
SISTEMA DISTRIBUÏT DE GESTIÓ DE TORNEJOS VIA WEB

TREBALL FINAL DE GRAU

BARCELONA, 31 DE GENER 2019



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

AUTOR
DIRECTOR

Xavier Peñalosa Esteller
Jordi Guitart Fernández

Resum

Les grans empreses sovint tenen problemes d'escal·labilitat i de migració quan treballen amb bases de dades i aplicacions web. En aquest treball es valoren les possibles sol·lucions a aquest problema analitzant diverses arquitectures i programaris que permetin agilitzar i simplificar el procés. S'implementa una sol·lució altament escal·lable, segura, fiable i ràpida d'instal·lar per a servidors de bases de dades basada en Apache ZooKeeper. Per a demostrar el seu funcionament en un entorn productiu, també es desenvolupa una aplicació web que permet gestionar les classificacions de diferents tornejos.

Abstract

Corporates and large companies are susceptible to scalability and migration problems when working with web-application databases. Several solutions are analyzed in this document, as well as the server architectures and software that allow for a simplified and swift process. A highly scalable, swift, secure and easy-to-install solution based on Apache ZooKeeper is implemented for the project and, in order to validate its correct functionality, a web-based application is developed. The web application allows for the management of classifications in tournament environments.

Index de continguts

1	Contextualització del projecte	1
1.1	Formulació del problema	1
1.1.1	Emmagatzematge de dades en arquitectures web	1
1.1.2	Aplicació web	1
1.1.3	Objectius	1
1.2	Abast	1
1.3	Metodologia i rigor	2
2	Estat de l'art	3
2.1	Context	3
2.1.1	Actors	3
2.2	Estat de l'art aplicat al projecte	4
2.2.1	Servidors distribuïts en xarxa local	4
2.2.2	Servidors distribuïts en xarxa pública	5
2.2.3	Programari de desenvolupament web	5
2.3	Ús de l'estat de l'art	5
3	Planificació	6
3.1	Tasques	6
3.2	Temps	8
3.3	Recursos	8
3.4	Esquema de Gantt	9
3.5	Valoració d'alternatives i pla d'acció	9
4	Pressupost	11
4.1	Reflexió	12
5	Informe de sostenibilitat	13
5.1	Impacte ambiental	13
5.2	Impacte econòmic	14
5.3	Impacte social	14
6	Desviacions respecte la planificació inicial	16
6.1	Objectius	16
6.2	Abast	16
6.3	Metodologia	16
6.4	Desviació temporal	16
6.5	Pressupost	17
7	Especificació tècnica	18
7.1	Arquitectura	18
7.1.1	Apache ZooKeeper	18
7.1.2	Instal·lació del programari	19
7.1.3	Model de dades	20
7.1.4	Operacions	20
7.1.5	Garanties de funcionament	20
7.2	Aplicació web	21
7.2.1	Apache	21
7.2.2	Django	21
7.2.3	Funcionalitats	22
7.2.4	Emmagatzematge de dades	23
7.3	Comunicacions	25
7.3.1	Connectivitat amb ZooKeeper	26
7.3.2	Condicions de carrera	27

8	Anàlisi del projecte	28
8.1	Proves manuals i automatitzades	28
8.2	Tolerància a fallades	29
8.3	Rendiment	29
9	Conclusions	33
	Bibliografia	34
	Annexos	

Glossari

Terme	Definició
API	Interfície de programació d'aplicacions, sigles de l'anglès <i>Application Programming Interface</i> .
Back-end	Part d'una aplicació web que s'encarrega d'emmagatzemar i llegir dades per al funcionament de l'aplicació.
Framework	Entorn de treball. Programari que ofereix múltiples funcionalitats relacionades amb el desenvolupament.
Front-end	Part d'una aplicació web que s'encarrega de rebre les peticions dels usuaris i mostrar les dades sol·licitades de forma visual.
HTML	Hypertext Markup Language, llenguatge usat per al desenvolupament d'aplicacions web.
HTTP	Hypertext Transfer Protocol, protocol per a l'intercanvi de documents d'hipertext i multimèdia al web.
Latència	Temps de resposta d'un servidor o una aplicació web.
Servidor esclau	En una arquitectura mestre-esclau, el servidor que duu a terme les peticions.
Servidor mestre	En una arquitectura mestre-esclau, el servidor que s'encarrega de delegar les peticions.
SVGA	Associació d'estudiants del Campus Nord (<i>Series and VideoGames Association</i>).
Throughput	Quantitat de dades servides per un sistema, sovint expressat en Kilobytes per segon.
TLD	Top-level domain. Domini web de primer nivell, per exemple <i>.com</i> , <i>.org</i> , ...
Uptime	Temps de disponibilitat d'un servei ofert, sovint una aplicació web. Expressat en percentatge.

1 Contextualització del projecte

1.1 Formulació del problema

Existeix una gran quantitat de programari per agrupar diferents servidors i aprofitar la potència de càlcul que ofereixen de forma conjunta. Tot i això, disposar de diferents servidors per emmagatzemar dades i servir-les o modificar-les a mida que es van rebent peticions externes és una pràctica que han adoptat les grans empreses per tal de garantir un temps de servei gairebé complert. L'objecte d'estudi d'aquest projecte es tracta de validar la viabilitat de diferents programaris per veure quin s'ajusta millor a arquitectures escal·lables, mantenint una configuració senzilla i sòlida, i centrant el focus d'atenció a la part d'emmagatzemament i accés de dades.

1.1.1 Emmagatzematge de dades en arquitectures web

Quan es desenvolupa un projecte d'aplicació web per a una gran empresa, és una pràctica comuna adaptar el funcionament de l'arquitectura a les necessitats del client. Es decideixen la quantitat de servidors que es volen usar i la funcionalitat que s'els dona, ja sigui com a base de dades, resposta a peticions web o com a sistema d'autenticació d'usuaris. Degut a la manca d'un sistema fàcilment escal·lable per a les bases de dades, sovint es repliquen els servidors dedicats a aquesta finalitat de forma manual. Aquest procés resulta lent quan es treballa amb bases de dades grans; en cas d'emergència, no tenir la capacitat per duplicar un servidor de dades pot suposar una pèrdua de diners i clients. Per tant, es troba una necessitat de tenir un sistema que actuï com a base de dades, distribuït en diferents servidors i que estigui estructurat en una arquitectura infinitament escal·lable.

1.1.2 Aplicació web

Aprofitant l'objecte d'estudi d'aquest projecte, s'ha decidit implementar una aplicació web que tingui un propòsit real, oferint així una eina de validació del funcionament de l'arquitectura. Des de l'associació juvenil del Campus Nord *SVGA* sovint s'organitzen competicions en forma de torneig. S'ha detectat que existeix una manca de plataformes gratuïtes que permetin gestionar les classificacions dels jugadors en un torneig. Les poques planes web que ofereixen aquest servei també requereixen que l'usuari es registri amb el seu compte de correu, sovint venent-lo a empreses publicitàries. Per tant, s'usarà el projecte per a implementar una aplicació web que ofereixi les funcionalitats descrites, de forma completament gratuïta i sense registres.

1.1.3 Objectius

Per considerar que el Treball Final de Grau s'ha finalitzat correctament, cal complir les següents fites:

1. Ha d'existir un sistema integrat per diversos servidors que es poden comunicar entre ells.
2. El sistema ha de ser tolerant a fallades dels seus components.
3. S'ha de mantenir la privacitat i consistència de les dades en tot moment.
4. El sistema ha de tenir com a mínim un punt de contacte amb Internet per transmetre les dades pertinents als usuaris interessats.
5. El sistema ha de servir una aplicació web.
6. L'aplicació web ha de servir per gestionar les classificacions dels diferents jugadors d'un torneig.

1.2 Abast

El projecte té com a objectiu dissenyar i implementar una arquitectura a diferents nivells que permet servir planes web. L'arquitectura resultant ha de complir amb les següents característiques per tal de ser viable en un àmbit professional.

- Alta tolerància a fallades de sistema.

- Alta escalabilitat.
- Consistència de dades en tot moment.
- *Uptime* superior al 95% del temps de sistema.
- Temps de resposta a peticions inferior als 3 segons en cas de càrrega de aplicació web. Resposta visual immediata en cas d'interacció amb aplicació web.

L'aplicació web oferirà un servei per a gestionar tornejos de temàtica genèrica i consultar les classificacions dels seus participants. Per tant, cal tenir un servei de gestió de dades que es pugui consultar i editar. Degut a la implementació amb múltiples servidors, s'idearà una forma de mantenir la consistència i coherència de les dades emmagatzemades. Per la part de web, els usuaris disposaran de diferents funcionalitats pròpies de l'àmbit del torneig. Hi ha funcionalitats que estaran limitades per contrasenya de forma que només les puguin fer servir els administradors del torneig. D'aquesta forma, s'evita que els jugadors accedeixin de forma il·lícita al torneig i modifiquin la classificació.

En cas que no es compleixi qualsevol dels requisits indicats, es valorarà per quin motiu no es pot obtenir la qualitat necessària i s'estudiaran futures vies de millora per al projecte. Els possibles problemes que es poden plantejar en aquest punt inicial de desenvolupament del projecte es relacionen amb la quantitat de peticions que pot respondre el servei de forma concurrent. Tot i que hi hagi diferents servidors per a mantenir les dades, hi ha d'haver un punt d'entrada al sistema, i és possible que aquest punt d'entrada actuï com a coll d'ampolla, limitant el nombre de peticions.

1.3 Metodologia i rigor

Es seguirà una metodologia àgil aplicada generalment al desenvolupament de software, la metodologia *Scrum*¹. Un dels múltiples avantatges que ofereix aquesta metodologia és la possibilitat de modificar els objectius a curt i mitjà termini sense haver de reorganitzar tota la planificació del projecte. Aquest fet permet tractar amb facilitat els imprevistos que puguin sorgir durant el procés de desenvolupament. El mètode es basa en definir les tasques principals del projecte i assignar unes puntuacions segons l'esforç que requereixen. Les tasques que tenen una puntuació prou alta s'han de redefinir per fer-les més granulars. D'aquesta forma es poden justificar les desviacions sobre les dates previstes de forma més precisa i es poden localitzar les tasques més problemàtiques.

Per tal de realitzar el seguiment de les tasques a desenvolupar i dels objectius assolits, s'ha creat un lloc web que actua com a registre d'accions. Aquest contingut es podrà accedir de forma complementària des de la web principal del projecte, tot i que no forma part ni de la web principal ni dels objectius del projecte. S'han definit unes tasques inicials que inclouen de forma general els objectius del projecte, amb una estimació de temps per a cada tasca. L'estimació de temps s'ha realitzat conforme amb la dificultat de la tasca i la rellevància que hi té en el projecte. Seguint la ideologia del mètode Scrum, les desviacions sobre les dates previstes s'afegiran al registre web, junt amb el motiu per el qual no es pot complir l'objectiu a temps. Un cop s'hagi completat el projecte, serà possible veure tot l'historial de desenvolupament junt amb les dates en les quals s'ha realitzat. Per tal de validar la finalització de les tasques es crearan objectius més granulars de forma que es puguin validar amb proves automatitzades via programari.

Adicionalment, es mantindran diverses reunions amb el director al llarg de la duració del projecte. Aquestes reunions tindran com a objectiu realitzar accions de seguiment i validació de les tasques desenvolupades, així com la valoració de temps i objectius assolibles.

¹Scrum - Viquipèdia. ca.wikipedia.org/w/index.php?title=Scrum&oldid=20224009

2 Estat de l'art

2.1 Context

Les planes web s'estan convertint en un estàndard d'accés a tot tipus d'informació. La ràpida informatització dels serveis des del desenvolupament d'internet ha creat una cursa a nivell mundial per oferir productes amb la millor qualitat possible. Per tal de satisfer les necessitats de tots els internautes, la quantitat de dominis web també creix anualment, tal com mostra la Figura 1. Cal remarcar que l'augment anual segueix una tendència gairebé exponencial, i actualment existeixen 1.700 milions de dominis actius a internet, un per a cada tres persones amb accés a internet (Netcraft, 2018).

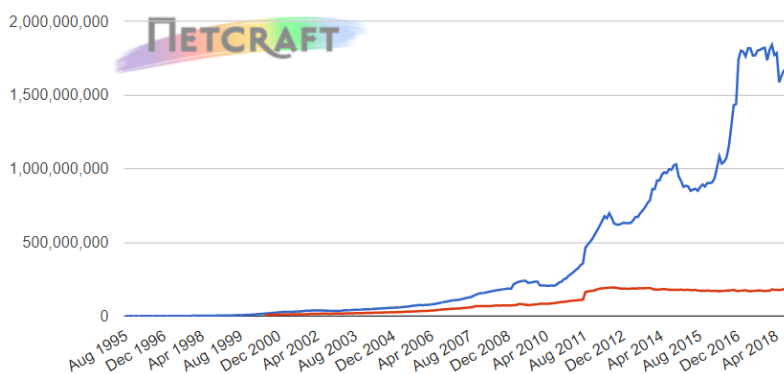


Figura 1: Quantitat d'elements web existents segons les estadístiques de Netcraft. En color blau, dominis actius. En color vermell, planes web actives.

La tecnologia associada al desenvolupament de llocs web pateix un renovament constant, tant per als llenguatges de desenvolupament com per a les plataformes que les serveixen. És per això que les arquitectures dels servidors darrere de planes web també s'han de renovar constantment si es vol oferir la millor qualitat de servei. Juntament amb el factor de qualitat, és important explorar les capacitats de càrrega que tenen les plataformes web. La capacitat de càrrega es pot definir com el màxim de connexions que pot suportar la plataforma de forma concurrent. Ja que aproximadament un 55% (Stats, 2018) de la població mundial disposa d'accés a internet, uns 4.200 milions de persones, cal tenir en compte que els llocs web han de ser capaços de suportar una gran quantitat de peticions, tot i que no necessàriament de forma concurrent.

Tal i com indica Daniel Mensacé (Menascé, 2002), la capacitat de càrrega sovint es troba limitada per l'arquitectura del sistema, els recursos que s'ocupen per respondre una petició i el temps que comporta construir i enviar la petició. Per augmentar la capacitat de càrrega, segueix Mensacé, es poden prendre diverses mesures tenint en compte els factors esmentats: Disminuir la *latència* o temps de resposta del servidor, optimitzar el contingut que es serveix des del servidor per que resulti més senzill construir la resposta a la petició, o augmentar els recursos de maquinari. En aquest projecte principalment s'avaluarà l'opció d'augmentar recursos de maquinari, però els altres factors no seran negligits ja que els resultats no serien apropiats per a un projecte escal·lable. L'escal·labilitat d'un projecte, concepte altament relacionat amb la capacitat de càrrega, es defineix com la capacitat de mantenir una quantitat elevada d'accessos sense perdre qualitat de resposta, i sovint s'aconsegueix mitjançant una millora via maquinari.

2.1.1 Actors

Les entitats que es poden beneficiar del projecte per diversos motius es troben llistades a continuació junt amb el seu possible interès.

- Entitats que es trobin amb la necessitat d'implementar una arquitectura escal·lable i a prova d'errors.
- Entitats que necessitin organitzar tornejos i no vulguin desenvolupar la seva pròpia eina, compartir informació personal o pagar un preu per l'ús de la plataforma.

- Entitats que estiguin desenvolupant una aplicació web amb objectiu similar i necessitin referències d'ús.
- L'autor del projecte, per ampliar els coneixements amb el disseny i l'implementació de l'arquitectura i l'aplicació web.

2.2 Estat de l'art aplicat al projecte

Per tal de valorar l'escal·labilitat de una arquitectura enfocada a servir pàgines web és necessari distingir entre dos blocs principals: Escal·labilitat mitjançant la millora dels servidors disponibles (vertical), i la escal·labilitat mitjançant l'augment del nombre de servidors disponibles (horitzontal) (Cardellini & Casalicchio, 2002). Degut a la limitació en la millora d'un únic servidor, ja sigui per components de maquinari o per limitacions del programari, resulta més senzill i eficaç centrar els esforços i el pressupost disponible en afegir més servidors a la plataforma (Henderson, 2006). L'enviament de dades entre els diferents servidors d'un sistema planteja dos blocs més, les arquitectures implementades dins d'una mateixa xarxa privada o local, i les arquitectures construïdes mitjançant comunicacions per la xarxa pública. Les primeres han pres molta força durant els darrers anys degut a la facilitat que proporcionen per configurar la seguretat de l'entorn, tot i que a vegades no és possible implementar-les degut a limitacions físiques o d'enrutat de xarxes.

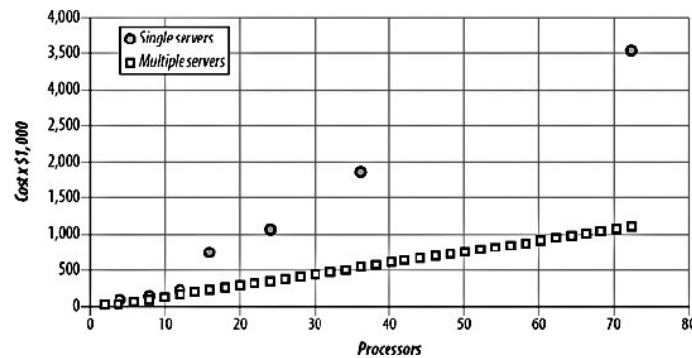


Figura 2: Relació de cost en milers de dòlars contra quantitat de processadors, usant escal·labilitat vertical (un servidor) i horitzontal (múltiples servidors)

2.2.1 Servidors distribuïts en xarxa local

Els servidors distribuïts en xarxa local són aquells que es transmeten informació dins d'un canal de comunicació privat. El canal de comunicació es pot generar dins d'una mateixa xarxa d'internet, connectada per un sol enrutador, o pot ser generada amb túnels TCP/IP o xarxes d'estil VPN (Xarxes privades virtuals). Aquests tipus de comunicacions permeten configurar diverses normes de tallafocs per impedir que hi hagi connexions externes no desitjades, evitant la necessitat d'encriptar les dades enviades entre els servidors.

Segons Valeria Cardellini i Emiliano Casalicchio (Cardellini & Casalicchio, 2002), els sistemes han d'oferir un sol punt de contacte amb l'exterior amb l'objectiu de facilitar les tasques d'enregistrament de connexions i resolució de problemes. Aquest punt de contacte és el que figura amb una direcció IP pública en els servidors *DNS* (Domain Name Server), que tenen la resolució del domini pertinent. El punt de contacte és l'encarregat de fer arribar les peticions a la resta de servidors, tot i que no necessàriament el que hi ofereix resposta. Degut a la problemàtica existent amb certs navegadors web que no són tolerants a aquest tipus de comportament, la sol·lució més habitual segons Özsü i Valdúriez (Özsü & Valdúriez, 2011) és implementar una arquitectura *mestre-esclau* on la base de dades sigui compartida de forma que totes les peticions tornin del mateix node però tots els nodes de l'arquitectura tinguin la informació per si hi ha cap fallada del sistema. En aquest tipus d'arquitectura, hi ha un servidor *mestre*, que actua com a distribuïdor de tasques i de comunicacions mentre que els altres servidors, denominats *esclaus*, reben les tasques i retornen els resultats al servidor mestre. Segons l'article de Cardellini (Cardellini & Casalicchio, 2002) i el llibre publicat per Andrew Tanenbaum i Maarten Van Steen (Tanenbaum & Van Steen, 2007), si els servidors esclau realitzen la mateixa tasca de forma intercanviable, reben el nom de *clúster*. En

canvi, si cada servidor esclau té una tasca específica es parla d'una arquitectura distribuïda. Els autors indiquen que els clústers són més comuns a l'àmbit de servei web i l'arquitectura distribuïda s'usa generalment per a aplicacions on es requereixen càlculs que es poden executar concurrentment i tenen un cost elevat.

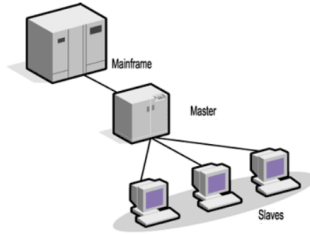


Figura 3: Exemple d'arquitectura Mestre-Esclau.

2.2.2 Servidors distribuïts en xarxa pública

Els servidors distribuïts en xarxa pública són aquells que es transmeten informació dins d'un canal de comunicació públic, generalment aprofitant l'estructura existent d'Internet. A nivell lògic el sistema actua de forma diferent que si s'implementés en una xarxa privada; existeixen diversos punts d'entrada al sistema i només es fa de forma privada la sincronització de les dades. Un exemple d'aquesta arquitectura són les criptomonedes² i el *BlockChain*³, que usen una tecnologia coneguda com *Distributed Ledger* (Llibre de comptes distribuït).

Segons l'article publicat per Svein Ølnes, Jolien Ubacht i Marijn Janssen (Ølnes, Ubacht, & Janssen, 2017), el fet d'aprofitar la xarxa pública per a transmetre dades implica que qualsevol connexió pot ser interceptada i modificada, comproment les dades que s'emmagatzemin als servidors. És per això que encriptar les dades enviades entre els servidors deixa de ser una opció i es converteix en una necessitat. Addicionalment, el sistema passa a tenir una dependència dels nodes intermedis, sobre els quals no es té cap control.

2.2.3 Programari de desenvolupament web

Existeixen moltes eines i plataformes per al desenvolupament de planes web, i es poden classificar segons el llenguatge de programació sobre el qual treballen. La majoria de llenguatges de programació amb suport per a planes web ofereixen un gran ventall de *frameworks* que consisteixen en codi precompilat amb funcionalitats d'autocompletat i una gran part dels fitxers de configuració ja preparats per a llençar una aplicació web. Julia Plekhanova (Plekhanova, 2009) afirma que els llenguatges més populars per al desenvolupament d'aplicacions web són PHP, Ruby i Python. Els frameworks associats també es comparen a l'explicació de Plekhanova i s'arriba a la conclusió que Django (Python) i Ruby on Rails (Ruby) són els millor valorats al món professional i els més senzills d'usar i aprendre.

2.3 Ús de l'estat de l'art

Un cop valorades totes les possibilitats que s'ofereixen, s'ha decidit implementar una arquitectura de xarxa privada tipus clúster amb comunicació mestre-esclau. Per tant, s'implementarà una sol·lució ja existent per donar suport a la plataforma web. Per tal de crear una plataforma a prova d'errades i simular un sistema escal·lable, s'adquiriran diversos servidors i es configuraran per a que treballin com a un sol conjunt. En cas que hi hagi cap servidor fora de la xarxa local, es configuraran les eines d'enrutament necessàries per tal de crear túnels segurs i establir una xarxa privada. L'eina de desenvolupament web serà Python, triat per la seva versatilitat i el coneixement previ del llenguatge per part de l'autor. Concretament, s'usarà la plataforma Django. Django ofereix un nivell de configuració prou acurat i un ventall d'esquemes predefinitos que permeten minimitzar el temps de desenvolupament web per centrar-lo en la implementació de l'arquitectura.

²Criptomonedes - Viquipèdia. ca.wikipedia.org/wiki/Criptomonedes

³BlockChain - Viquipèdia. ca.wikipedia.org/wiki/Blockchain

3 Planificació

Per tal d'estructurar bé el projecte, cal saber identificar de quina manera es pot subdividir en tasques. Les tasques ajuden a visualitzar els diferents objectius dels que consta un projecte, i ajuden a realitzar una estimació del temps a dedicar al projecte. Si es valoren els diferents obstacles que poden tenir les tasques de forma individual, es poden tenir en compte tots els obstacles que poden aparèixer durant el desenvolupament. Això permet reestructurar la planificació del projecte i valorar si cal modificar cap aspecte fins i tot abans de començar.

3.1 Tasques

Les diferents tasques del projecte s'han agrupat seguint un criteri de similitud per tal de poder identificar més clarament en quin punt de desenvolupament es troba el projecte. Agrupar les tasques segons aquest criteri permet identificar els problemes que poden sorgir i actuar en conseqüència cedint més hores a les tasques més problemàtiques.

Identificador	Descripció	Estimació de temps (hores)	Dependències
T1	Investigació prèvia	30	-
T2	Especificació de projecte	20	-
T3	Recapte de recursos	20	T2
T4	Creació de registre web	20	-
T5	Configuració de servidors	30	T3
T6	Gestió de projectes (GEP)	70	-
T7	Desenvolupament d'arquitectura	60	T2, T5
T8	Desenvolupament de web	40	T7
T9	Proves internes	30	-
T10	Proves externes	20	T9
-	Total	340	-

Taula 1: Tasques principals a dur a terme, duració estimada i dependències.

Exceptuant les tasques T2, T3, T9 i T10, totes les tasques es divideixen en subtasques degut a la seva complexitat. Tot i que no es troba representat a la taula 1, totes les tasques tenen una dependència implícita amb la tasca T1, ja que és el pas previ al desenvolupament del projecte. Les tasques es troben definides a continuació amb una breu explicació dels seus objectius i les seves subtasques.

T1 - Investigació prèvia

L'investigació prèvia és un primer pas essencial en el desenvolupament de qualsevol projecte i sovint aquesta tasca la desenvolupa el Gestor de Projectes. Cal validar que no existeixen projectes amb la mateixa funcionalitat. Si n'existeixen, cal veure si satisfà les necessitats dels usuaris i si es pot millorar o arreglar de cap forma. En aquest apartat s'estudiarà la viabilitat del projecte a realitzar i l'estat de l'art amb la tecnologia que es vol usar. Un cop s'hagin explorat les diferents opcions, es triarà el programari adient o es valorarà si cal desenvolupar nou programari per tal de realitzar el projecte.

T2 - Especificació de projecte

Un cop s'ha triat el programari i el maquinari que es vol usar o desenvolupar, cal plantejar de quina manera es vol combinar. Aquesta tasca serveix per planificar el projecte, definir les tasques a fer i establir un mètode de treball; tasques pròpies d'un Director Tècnic. Tenint aquesta informació es pot procedir a fer una estimació dels costos temporals i els econòmics dels materials necessaris per al projecte junt amb Recursos Humans. Si es veu que el cost econòmic supera els valors màxims establerts per els gestors del projecte, es pot realitzar una altra iteració per a decidir quines eines es poden adaptar millor al pressupost.

T3 - Recapte de recursos

Havent especificat els materials necessaris per a desenvolupar el projecte, cal començar a adquirir-los si no es disposa d'ells. Els materials físics com servidors o enrutadors es poden adquirir mitjançant una compra, un lloguer o un préstec. Depenent de la opció triada, es pot trigar un cert temps des de que es fa la sol·licitud fins que es reben els materials. Per al que fa el programari, la seva compra o descàrrega comporta una adquisició immediata; el seu desenvolupament, si és necessari, generalment forma part de les tasques principals del projecte.

T4 - Creació de registre web

Per tal de realitzar un seguiment adient del projecte és necessari usar una eina que permeti anotar el progrés. Actualment existeixen una gran varietat de serveis en línia que ofereixen aquesta possibilitat, des d'editors de text senzills com els que ofereixen Google Docs⁴ fins a eines més avançades i especialitzades com Trello⁵. Ja que s'ha triat usar la plataforma Django per al desenvolupament de l'aplicació web del projecte, s'ha optat per desenvolupar una eina senzilla de forma que l'autor del projecte pugui aprendre conceptes bàsics del funcionament de Django. Per tal de desenvolupar aquesta eina caldrà realitzar una configuració inicial del servidor i desenvolupar les funcionalitats bàsiques d'accés i modificació de dades.

T5 - Configuració de servidors

La tasca de configuració de servidors és primordial per al correcte desenvolupament i funcionament del projecte. Aquesta tasca consisteix en configurar l'enrutat de les vies de comunicació per tal de que els diferents servidors es puguin comunicar entre ells, compartint les dades i les peticions que els arriben. Amb aquest objectiu, cal configurar el programari necessari i instal·lar el programari que hi manca. Addicionalment en aquesta tasca s'afegiran una sèrie de proves d'integració que validin la visibilitat entre els servidors.

T6 - Gestió de projectes (GEP)

Cal tenir en compte el temps de dedicació a l'assignatura de Gestió de Projectes, ja que forma part del desenvolupament i documentació del projecte. La tasca es pot dividir en els diferents lliuraments que s'han de realitzar. Degut a la guia docent es pot realitzar una estimació bastant acurada del temps que s'haurà de dedicar a aquesta tasca.

T7 - Desenvolupament d'arquitectura

La tasca principal del projecte, consisteix en organitzar l'arquitectura que ha de donar suport a l'aplicació web. S'ha de dissenyar el model de dades que es vol fer servir i el mètode que s'usarà per accedir-hi. Es desenvoluparà o s'implementarà una eina que ofereixi el protocol de comunicació entre els diferents servidors i es validarà que funcioni per al tipus de dades que s'han modelat. Un cop la comunicació sigui eficaç es desenvoluparan una sèrie de tests que permetin identificar amb rapidesa si hi ha cap error i indicar quin apartat de l'arquitectura n'és el responsable. Amb l'accés a dades verificat, caldrà crear una *API* que pugui gestionar les peticions de sol·licitud i/o modificació de dades, de manera que es puguin accedir des de qualsevol aplicació web independentment de la seva temàtica. Aquesta tasca ha de ser desenvolupada per el departament tècnic amb la supervisió del director tècnic i el director de projecte degut a la seva importància.

T8 - Desenvolupament de web

El desenvolupament de l'aplicació web és la part més visual del projecte. Tot i això, en el projecte es valorarà que tingui un funcionament correcte per sobre del resultat artístic que pugui presentar. Es desenvoluparà una aplicació web que permeti accedir i modificar les dades emmagatzemades en els servidors de forma que pugui oferir el servei de gestió de tornejos decidit inicialment. Per tal de permetre l'accés des de diferents dispositius es crearà una API amb les funcionalitats bàsiques de la web. La web estarà composta de les dues parts típiques en arquitectura web, el *front-end* i el *back-end*. Un cop creada la API, que serà la part d'accés a dades o back-end, es centraran els

⁴Google Docs. Desenvolupat per Google, 2007. docs.google.com

⁵Trello. Desenvolupat per Fog Creek Software, 2011. trello.com

esforços en desenvolupar una part visual o front-end. Es podria dedicar una part especialitzada de l'equip de desenvolupament per a que tracti amb el desenvolupament web, tot i que en aquest Treball Final de Grau l'aplicació web no és l'objectiu principal.

T9 - Proves internes

Un cop finalitzat el desenvolupament del projecte, es dedicarà un conjunt d'hores a verificar que totes les funcionalitats actuen correctament, i que no hi ha cap problema de connectivitat o de gestió del sistema. Per aconseguir aquestes fites l'equip de Qualitat generarà un grup de proves que revisin exhaustivament totes les funcionalitats. En cas que es trobi cap errada, el temps de resolució de problema s'inclourà dins d'aquesta mateixa tasca i es traslladarà a l'equip tècnic.

T10 - Proves externes

Una part molt important de qualsevol projecte és la part de proves externes. Els desenvolupadors que dissenyen el projecte tenen identificats una sèrie de casos d'ús, generalment adaptats a les necessitats bàsiques dels projectes i per tant és possible que passin per alt casos d'ús extrems o activitats irregulars. Mostrar una primera versió del projecte a persones externes al seu desenvolupament assegura que en certa mida aquests casos d'ús es vegin reduïts. Amb l'objectiu d'observar els comportaments de diferents usuaris davant del projecte, s'ideen un conjunt de proves senzilles per dur a terme sense coneixement de l'aplicació web o l'arquitectura. Els resultats s'analitzen per tal de trobar mancances o errors, i es recullen les opinions dels usuaris per veure què es podria millorar a nivell d'usabilitat. Si és possible, cal obtenir voluntaris representatius dels diferents usuaris finals que poden usar l'aplicació web amb l'objectiu de recollir opinions variades sobre el seu funcionament.

3.2 Temps

S'assumeix que hi haurà una dedicació diària d'entre 2 i 3 hores per als dies laborables, i entre 6 i 8 hores els dies no laborables. Les hores que es poden dedicar al projecte als dies laborables són menors degut a la vida professional de l'autor fora de la universitat. Per tal de fer una aproximació del cost temporal de les tasques s'ha tingut en compte la dificultat de les mateixes i la rellevància que tenen dins el projecte. Fent el recompte de les hores dedicades a cada tasca s'obté un total de 340 hores (Taula 1) que s'acosta al valor de 375 hores, extret del càlcul dels 15 crèdits ECTS del Treball Final de Grau a 25 hores per crèdit. Les hores restants es deixen com a hores de coixí per si surt cap imprevist durant el desenvolupament.

La tasca T10 (Proves externes) requereix voluntaris externs al desenvolupament del projecte, i és possible que no tinguin disponibilitat per a realitzar les proves especificades quan l'autor hagi especificat. Per tant, es deixa un període major al mínim necessari degut als possibles problemes de disponibilitat dels voluntaris. També cal tenir en compte que es realitzaran diverses iteracions de millora i proves en funció de les opinions dels usuaris.

3.3 Recursos

Els recursos materials necessaris per al desenvolupament del projecte són els llistats a continuació, i els seus costos econòmics associats es poden veure detallats a la Secció 4. Per el que fa als recursos temporals, es poden veure reflectits a la Taula 1 i a la Secció 3.2.

- Servidors (3 o més) per a l'arquitectura del projecte.
- Enrutador per a la connectivitat.
- Connexió a internet per servir l'aplicació web.
- Ordinador per a desenvolupar el projecte.
- Editor de text⁶ per a desenvolupar el projecte.
- Energia (Electricitat) per al funcionament de tots els aparells electrònics del projecte.

⁶Vim. Desenvolupat per Bram Moolenaar, 1991. www.vim.org

- Desenvolupador (1).
- Lloc físic per a emmagatzemar els servidors i enrutadors.

Distribució de rols

Tractant-se d'un projecte relativament petit s'ha considerat que no cal contractar més personal per que desenvolupin els rols com Gestor de Projecte i Recursos Humans, i tampoc cal contractar més desenvolupadors. En conseqüència, l'autor del projecte s'encarregarà de tots els rols del projecte indiferentment de les tasques que s'han de realitzar.

3.4 Esquema de Gantt

Un cop establerta la duració en hores de les tasques a realitzar, cal organitzar-les segons l'ordre lògic a seguir. Les tasques que depenguin d'altres tasques es planificaran per moments posteriors, quan es puguin completar sense restriccions. La relació entre les diferents tasques i l'esquema de Gantt planificat es poden veure a l'Annex A.

Les diferents fites del projecte s'identifiquen amb una separació vertical de color vermell a l'esquema de Gantt de l'annex. Aquestes fites indiquen un canvi en les accions de desenvolupament del projecte i estan fortament lligades amb les tasques a desenvolupar.

1. Fita d'investigació - Inicis de Setembre. Indica la finalització de les tasques d'investigació del projecte.
2. Fita de configuració - Finals d'Octubre. Indica la finalització de les tasques de comunicació entre els diferents servidors.
3. Fita de desenvolupament - Finals de Novembre. Es considera assolida quan el desenvolupament del projecte està arribant al seu final i s'ha de començar la validació de totes les parts.
4. Fita final - Finals de Desembre o principis de Gener. Indica la finalització del projecte i la presentació del Treball Final de Grau davant del tribunal.

Dependències

Les dependències es troben marcades a l'esquema de Gantt mitjançant connectors que enllacen el final de la tasca inicial amb el començament de la tasca depenent, exceptuant la tasca T1 que s'hauria d'enllaçar amb totes les tasques i reduiria la llegibilitat de l'esquema. La Taula 2 mostra els diferents motius de cada dependència entre les tasques.

3.5 Valoració d'alternatives i pla d'acció

Les petites desviacions temporals són inevitables en tots els grans projectes. És essencial saber com enfrontar-s'hi i estar preparat per a re-estructurar la planificació original. Els problemes que s'han identificat en relació a les tasques planificades són els següents.

- Manca de material (Servidors, enrutadors).
- Impossibilitat de configurar la comunicació entre diversos elements.
- Invalidesa del model de dades.
- Errades de configuració.
- Errades de llibreries i/o programari.
- Manca de voluntaris per a les proves externes.

Identificador	Dependència	Motiu
Totes	T1	La investigació prèvia és necessària per valorar la viabilitat del projecte i les eines que es poden usar.
T3	T2	Cal estudiar quins recursos es necessiten en el projecte abans d'iniciar la seva adquisició.
T5	T3	Per poder configurar la connectivitat dels servidors i el programari necessari per a la seva comunicació cal disposar del material.
T7	T2	Una bona especificació de projecte evita realitzar gaire iteracions durant el desenvolupament i controla els imprevistos que puguin aparèixer.
T7	T5	No es poden començar a implementar models de dades i de comunicació si els diferents elements de l'arquitectura no es poden comunicar entre ells.
T8	T7	L'aplicació web requereix tenir una arquitectura al darrere que pugui suportar les seves peticions i emmagatzemar les dades necessàries.
T10	T9	Les proves externes requereixen que tot el procés de validació interna hagi finalitzat. Tot i que es poden modificar funcionalitats de forma superficial, aquesta tasca té per objectiu trobar errades que hagin passat per alt al desenvolupador.

Taula 2: Dependències entre tasques amb els motius de dependència. Els identificadors es poden consultar a la Secció 3.1.

Alternatives

La manca de material és el problema principal, ja que sense el material adient no es pot dur a terme el projecte. Des del començament del projecte s'han començat a adquirir els elements necessaris per al desenvolupament, i es preveu que es pugui disposar de tots els elements abans de la finalització del temps dedicat a la tasca.

Per a les possibles errades de configuració i de programari s'ha inclòs una part de creació de proves a cada tasca. Aquestes proves tenen com a objectiu detectar de forma immediata quina secció del codi o de la configuració falla. Per tant, els errors induïts per aquests problemes haurien de ser gairebé negligibles.

La manca de voluntaris per a les proves externes és un factor social i és difícil de valorar quines alternatives hi poden haver. En cas que el nombre de voluntaris no sigui suficient per tenir una revisió de qualitat, es restaran hores de dedicació a la tasca per tal de poder polir altres aspectes del projecte. En el cas de la planificació del projecte, les desviacions es poden tractar mitjançant el grup d'hores no assignades a cap tasca; la planificació original disposa de 35 hores extres respecte a la càrrega de treball que haurien de comportar els 15 crèdits ECTS de l'assignatura. En cas que aquestes 35 hores no fossin suficients per a solventar els problemes que hagin causat les desviacions, s'escurçaria la tasca T10 (Proves externes) per donar lloc a les resolucions oportunes. Si tampoc fos possible encabir la resolució dels problemes dins del termini de lliurament del Treball Final de Grau, es valoraria amb el tutor del projecte quins objectius secundaris es poden reduir o obviar per tal d'aconseguir finalitzar-lo a temps. En cas que no es poguessin eliminar prou objectius, s'hauria de demanar de forma oficial a la universitat una ampliació del termini d'entrega.

Consum de recursos

Degut a que el projecte es tracta de la implementació d'una aplicació web amb l'arquitectura necessària per suportar-la, l'augment dels recursos necessaris en funció del temps és relativament menor. Un cop adquirits tots els elements necessaris només cal tenir en compte el consum d'electricitat i el temps de desenvolupament per part de l'autor. L'augment d'aquests dos elements es tradueix directament a un augment del cost econòmic del projecte (Veure Seccions 4 i 5.1).

4 Pressupost

Cal tenir en compte que el projecte s'ha comptabilitzat amb un cost temporal de 340 hores. Traduint el conjunt d'hores a una jornada laboral de 8 hores diàries s'obté una duració de 9 setmanes, o 2 mesos i una setmana. Els elements que tenen un cost mensual o diari s'adaptaran a aquesta xifra per a calcular el cost total, arrodonint cap a la fita superior. Per tal de calcular els costos econòmics del projecte cal especificar els recursos necessaris per al desenvolupament del projecte. En el següent llistat només s'indiquen els recursos amb un cost associat, independentment de si es tracten de recursos materials o temporals.

- Servidors (Raspberry Pi Zero W).
- Connectors per als servidors (Presa de corrent, tarja microSD).
- Enrutador (Cisco EPC3825).
- Connexió a internet (ADSL Ono).
- Domini web (TLD .com).
- Ordinador (LENOVO IdeaPad 330S).
- Electricitat.
- Enllumenat (Lamp.CorePro LEDtube 16W 840)
- Sou per al desenvolupador.
- Lloc físic per a emmagatzemar els servidors i enrutadors ($30m^2$).
- Mobiliari per a treballar dins del local (Taula, cadires, endolls)

Per calcular les amortitzacions dels elements pertinents s'aplicaran les amortitzacions lineals indicades per l'Agència Tributària durant el període 2017-2018⁷ degut a la manca de dades més actualitzades. S'assumeix que la vida útil dels elements informàtics és de 5 anys i per tant es pot aplicar un 20% d'amortització cada any sobre el preu total. Tot i que els elements amb valor unitari inferior als 601,01 euros no es poden amortitzar, s'ha afegit el seu cost a la Taula 3. Els elements amortitzables s'han de usar durant més temps del que es dedicarà al projecte degut a les limitacions legals de les amortitzacions. Per tant, tant els servidors com els enrutadors i l'ordinador es podran usar en altres projectes, reduint així els futurs costos econòmics.

El sou del desenvolupador serà de 10 euros/hora. Aquest preu s'obté a partir de la baixa experiència de l'autor en desenvolupament web i la seva situació d'estudiant. Si el projecte fos desenvolupat per un expert en el camp es reduiria el nombre d'hores necessaries per a implementar-lo; el mateix valor del projecte en menys temps augmentaria el valor del desenvolupador i en conseqüència el seu sou. Per tant, s'assumeix el rol d'un estudiant sense experiència laboral en el sector.

Tal com mostra la Taula 3, el cost total del projecte és de 5629,61 euros sense tenir en compte els costos de transport de material fins al local o les taxes aplicables als elements pertinents. La inversió inicial per dur a terme el projecte es pot reduir fins als 1346,32 euros si es descompten el sou i els costos mensuals de llum, connexió a internet i lloguer del local. En cas que fos necessari ampliar el termini de finalització del projecte degut a imprevistos s'hauria d'afegir al cost total els imports dependents de temps com el sou o l'electricitat. Si el motiu d'ampliació fos degut a una errada de maquinari s'hauria de valorar si és adequat adquirir un element de substitució. Degut a la curta durada del projecte, es pot assumir que els elements principals es troben dins del període de garantia i no caldria pagar el preu complet per a cap element. Per tant, es pot afegir un 10% del pressupost (550 euros) com a cost de contingència.

Per calcular el cost del local s'ha extret el valor del lloguer mitjà a la zona de Barcelona (14 euros/ m^2) aplicat als $30m^2$ de superfície i els 3 mesos de desenvolupament de projecte (degut a la impossibilitat de llogar un local per períodes menors a un mes natural).

⁷www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/eriodos_impositivos_a_partir_de_1_1_2017/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml

Concepte	Quantitat	Cost (Euros)		Amortització (Euros)	
		Unitari	Total	Anual	Mensual
Servidor	3	8,70	26,10	5,22	0,44
Material servidor	3	26,06	78,18	15,64	1,31
Enrutador	1	26,10	26,10	5,22	0,43
Connexió internet	3 (mesos)	18,15	54,45		
Domini web	1	15,99	15,99		
Ordinador	1	699,90	699,90	139,98	11,66
Electricitat	51,85 (kW)*	0,135	6,99		
Enllumenat	2	10,95	21,90	2,19	0,18
Sou	340 (hores)	10,00	3400,00		
Local	3 (mesos)	420,00 mensual	1260,00		
Mobiliari	1	40,00	40,00	4,00	0,33
Total			5629,61	172,25	14,35

Taula 3: Relació de materials i costos associats al projecte. *Veure Secció 5.1.

4.1 Reflexió

L'estat de l'art usat en les implementacions d'arquitectura similars a la d'aquest projecte normalment consten de servidors dedicats, amb molta potència de càlcul i bases de dades. Normalment aquests projectes es desenvolupen per mitjanes i grans empreses ja que garanteix un grau de redundància necessari per a servir planes web a gran escala, i els recursos econòmics dels que es disposen són molt majors als que es poden usar per a un Treball Final de Grau. És per això que s'ha intentat minimitzar el cost del projecte mitjançant la compra d'elements menys potents com les Raspberry Pi i els enrutadors domèstics de Cisco. Per una altra banda, un cop desenvolupat i gràcies a la seva escalabilitat, el projecte es podrà traslladar a servidors amb més recursos de forma que es puguin servir més peticions. Per tant, el cost de desenvolupament del projecte es tracta d'una part necessària però més econòmica que la seva implementació i manteniment en una arquitectura adequada per a les empreses que el poden necessitar.

Cal remarcar que els elements que comporten un cost major són el lloguer del local, el sou del desenvolupador i la compra de l'ordinador per a desenvolupar el projecte. És possible reduir el lloguer del local si es lloga en una zona on el cost per metre quadrat sigui inferior, i el cost del portàtil es pot reduir mitjançant la compra d'un portàtil de gamma baixa.

Per tal d'obtenir beneficis a partir d'aquest projecte es podria oferir la sol·lució en forma de programari a certes empreses que busquin un tipus d'arquitectura similar. Es podria escalar el preu d'implementació al sou d'un tècnic sènior i retallar els costos del material. Assumint que es pogués vendre el projecte per valor de 1500 euros, un preu molt inferior al sou d'un empleat tècnic, s'obtindrien beneficis amb només 4 vendes a diferents empreses. Si addicionalment s'oferís un servei de manteniment i suport es podria augmentar el benefici de forma considerable.

5 Informe de sostenibilitat

En aquesta secció s'analitza l'impacte ambiental, econòmic i social que té el projecte durant el seu desenvolupament i la seva vida útil. A cada apartat es valoren els possibles riscos que pot tenir el projecte per a l'àmbit, i es proposen sol·lucions adients. El conjunt de les següents seccions dona resposta a la matriu de sostenibilitat requerida, però en un format més extens.

5.1 Impacte ambiental

Projecte posat en producció

En relació al valor de kW/h per l'entrada d'Electricitat a la Taula 3, s'ha tingut en compte el cost de l'electricitat consumida per els servidors i els enrutadors durant el període actiu, i el cost de l'enllumenat necessari per a il·luminar el local.

- Segons RasPi.TV⁸, el cost energètic d'una Raspberry Pi Zero W és de 0,51Wh.
- Segons les especificacions de l'enrutador Cisco EPC3825, aquest té un consum de 18Wh.
- Assumint que per il·luminar el local són suficients 2 tubs LED del tipus especificat al llistat de la Secció 4 amb un consum de 16Wh per a cada un, s'obté un consum de 32Wh en conjunt.
- Segons les especificacions del fabricant, la bateria de l'ordinador indicat té un consum de 30Wh.

Per tant, s'obté un consum de 81,53Wh durant hores de desenvolupament i un consum de 19,53Wh durant les hores de descans. Seguint la planificació temporal de desenvolupament durant les 8 hores de jornada laboral, es calcula un consum de 964,72W els dies laborables i 468,72W els dies no laborables (5,76kW setmanals) i en conseqüència un consum total de 51,85kW per a les 9 setmanes que ha de durar el projecte. Aquest consum es podria reduir mitjançant el lloguer d'un local ben il·luminat de forma natural i apagant els elements tecnològics fora de les hores de desenvolupament. Degut als motius exposats a la Secció 4.1 es pot deduir que el desenvolupament del projecte amb material més econòmic i amb menys recursos també comporta una reducció del consum elèctric dels diferents components respecte a les implementacions habituals de l'estat de l'art en l'arquitectura usada.

S'hauria d'incloure el cost ambiental que suposa la recollida de materials i minerals per a poder construir els diferents elements tecnològics que conformen aquest projecte, així com les emissions de CO₂ del procés de fabricació i transport, però les empreses distribuïdores no fan pública aquesta informació i per tant no és calculable.

Vida útil

Ja que no cal mantenir l'enllumenat encès durant la vida útil del projecte, només cal tenir en compte el consum elèctric dels servidors i els enrutadors que proporcionen l'accés a l'aplicació web (19,53Wh). Assumint que tots els elements es troben encesos de forma permanent, el consum diari és de 468,72W i el setmanal de 3,21kW. Comparant aquest valor contra l'obtingut a l'apartat anterior de 51,85kW durant el desenvolupament del projecte, veiem que representa un 6,3% del consum. Aquest valor ens indica que durant la vida útil del projecte, aquest tindrà una empremta ecològica molt menor que la obtinguda durant el desenvolupament.

Riscos

L'empremta ecològica del projecte es podria veure augmentada si durant el desenvolupament o la vida útil hi apareix material defectuós. Aquest material s'hauria de substituir, augmentant el cost de recollida, processat i transport de les matèries primeres i en conseqüència les emissions de CO₂ introduïdes per tot el procés.

⁸raspi.tv/2017/how-much-power-does-pi-zero-w-use

5.2 Impacte econòmic

Projecte posat en producció

L'anàlisi dels costos per a la posada en producció del projecte s'expliciten a la Secció 4. Una inversió inicial de 1346,32 euros és un cost relativament baix, i no hi hauria cap problema per a trobar un inversor que facilités aquests diners.

S'han emprès mesures per a reduir el cost del projecte, com limitar la quantitat de servidors que s'usen per al desenvolupament o el personal dedicat al projecte. Si s'hagués contractat més personal per a fer la feina del projecte caldria tenir en compte els seus sous de cara al cost total, i aquest seria molt superior encara que la duració del projecte es veiés altament reduïda. Cal remarcar que el cost real del projecte ha sigut una mica superior a l'estimat inicialment, degut a diversos factors que s'expliciten a la Secció 6. Tot i això, la quantitat entra dins del pressupost destinat a la contingència d'emergències indicada a la Secció 4.1.

Vida útil

El projecte vol ser una alternativa segura i ràpida de desplegar per a les grans empreses. El temps destinat per un equip de professionals a perfilar l'arquitectura es pot trobar entre 2 i 3 setmanes; aquest temps comporta un sou per a cada integrant de l'equip. La sol·lució que ofereix aquest projecte per a l'arquitectura dels servidors amb bases de dades suposa una reducció d'aquest temps fins a un o dos dies. Aquest canvi suposa una reducció dels costos que ha de suportar l'empresa, tant temporals com econòmics.

És inevitable haver de comprar els servidors físics per al desplegament de l'arquitectura, i per tant és impossible reduir els costos del material i el seu manteniment. El que s'ofereix és una sol·lució ràpida i infinitament escal·lable; els clients de l'empresa podran accedir al contingut amb una disposició del 100%, evitant que les vendes i la imatge de l'empresa es vegin afectades.

Riscos

En cas que no es trobés un inversor inicial per a començar el desenvolupament del projecte, aquest es podria veure compromès en la seva realització. El desenvolupador hauria de fer la inversió inicial i no podria obtenir beneficis fins la venda del producte final. També cal tenir en compte els diferents elements del projecte que tenen un cost econòmic associat; si aquest cost puja per motius aliens o si cal afegir el cost de reparació o renovació, el desenvolupador es veuria directament afectat.

5.3 Impacte social

Projecte posat en producció

A nivell personal, aquest projecte aporta una millora directa dels coneixements sobre gestió de comunicacions i del sistema operatiu, així com de la programació en llenguatge Python i en menor mesura llenguatge web. Addicionalment, el projecte ofereix la possibilitat d'aprendre com funcionen els entorns de sistemes com Django i Apache, eines molt comunes al món professional. També és molt bona pràctica de cara al futur professional per fer estimacions de pressupost i temps d'un projecte, i es destaquen els punts febles a tenir en compte.

Vida útil

Degut a la naturalesa del projecte com a eina d'automatització de tasques, és possible que tingui una repercussió social negativa per als desenvolupadors. Els negocis que vulguin implementar una aplicació web i decideixin usar aquest Treball Final de Grau com a arquitectura no necessitaran contractar un enginyer informàtic per a que construeixi la estructura de servidors, només que s'encarregui del manteniment. De forma efectiva, els costos per al negoci disminueixen notablement. En conseqüència, el temps necessari per a la implementació de l'arquitectura també serà molt menor, minvant el temps d'implementació del projecte d'aplicació web. Degut a la redundància que ofereix aquest projecte, també es veu positivament afectat el temps de disponibilitat dels negocis i pot oferir un avantatge sobre la competència. Per una altra banda, el projecte és beneficiós per a l'entitat *SVGA* degut a la necessitat del tipus d'aplicació web amb oferta de classificació de tornejos.

Riscs

Com qualsevol projecte informàtic crucial per a un sistema, és possible que es creï una dependència a llarg termini. Aquest és un problema amb el qual es troben la majoria de grans empreses; apareix una versió millor del programari al mercat i cal valorar si es vol traslladar tot el sistema o si es vol seguir treballant amb la versió actual.

6 Desviacions respecte la planificació inicial

Es detecta una desviació temporal i econòmica respecte la planificació inicial en realitzar la fita de seguiment, però s'aconsegueix contenir la desviació per a la fita final sense modificacions addicionals. Per tant, només cal calcular els costos que suposen per al projecte les desviacions introduïdes anteriorment a la fita de seguiment.

6.1 Objectius

No hi ha cap desviació respecte als objectius especificats inicialment, i aquests s'han complert satisfactòriament en el projecte.

1. Existeix un sistema integrat per diversos servidors que es poden comunicar entre ells.
2. El sistema és tolerant a fallades dels seus components.
3. Es manté la privacitat i consistència de les dades en tot moment.
4. El sistema té un punt de contacte amb Internet per transmetre les dades pertinents als usuaris interessats.
5. El sistema serveix una aplicació web.
6. L'aplicació web permet gestionar les classificacions dels diferents jugadors d'un torneig.

6.2 Abast

Existeix una modificació respecte a l'abast del projecte planificat inicialment; No és possible triar la modalitat d'eliminació dels jugadors en el moment de crear el torneig. El model de dades suporta els diferents tipus de modalitats però degut al desconeixement de l'autor en desenvolupament de planes web, en aquest projecte no ha resultat possible representar-les mitjançant HTML. S'ha optat per ocultar el camp en el moment de creació del torneig per tal d'evitar la possible confusió dels usuaris.

6.3 Metodologia

S'ha introduït un canvi en la metodologia proposada inicialment degut als avantatges que suposa. El desenvolupament del programari es refina per a ser *desenvolupament guiat per proves*⁹, una pràctica de programació que es basa en definir les proves per al funcionament del programari abans d'implementar el codi. D'aquesta forma es pot verificar que les funcionalitats requerides es compleixen a mida que es va desenvolupant el programari. Combinant aquesta pràctica amb la metodologia Scrum (Secció 1.3) s'assoleix una eficiència molt superior, reduint gairebé a nul el temps de proves i modificacions posterior a la programació del codi.

6.4 Desviació temporal

Les desviacions respecte a la planificació inicial són relativament menors, i permeten reduir certs costos associats al desenvolupament del projecte. La tasca T3 (Recapte de recursos) es veu endarrerida 2 setmanes degut al préstec de dos servidors Raspberry Pi i el material necessari per al seu ús, i la tasca T7 (Desenvolupament d'arquitectura) es veu ampliada amb un temps equivalent a dues setmanes degut a la dificultat que presenta i una pobre estimació inicial de l'esforç a dedicar.

Aquests desviaments desencadenen una modificació a la resta de tasques posteriors, i per tant aquestes es troben desplaçades temporalment de la següent forma:

- T5 - Desplaçada 5 setmanes degut a la immobilitat de la tasca T6.
- T6 - No es pot desplaçar degut a les dates límit de les entregues.
- T7 - Desplaçada 2 setmanes per poder completar la tasca T5, augment de la duració en 2 setmanes.

⁹Desenvolupament guiat per proves - Viquipèdia. ca.wikipedia.org/wiki/Desenvolupament_guiat_per_proves

- T8 - Desplaçada 2 setmanes per la tasca T5, i 2 setmanes més degut a l'augment d'hores per a la tasca T7.
- T9 - Degut al canvi en el mètode de treball indicat a la Secció 6.3, es redueix en una setmana el temps dedicat a la tasca, i es pot començar conjuntament amb la tasca T8.
- T10 - Desplaçada 2 setmanes per la tasca T5. Les dues setmanes introduïdes per la tasca T7 es cancel·len amb la tasca T9.

Adicionalment, a causa de les desviacions indicades, cal redefinir la localització de les fites indicades a la Secció 3.4. Aquestes es desplacen al moment de finalització de les tasques adients. L'esquema de Gantt actualitzat per reflectir els canvis introduïts es pot trobar a l'Annex A.

6.5 Pressupost

Tant les desviacions temporals com el mètode d'adquisició del material impliquen una modificació dels costos del projecte.

- El cost dels servidors i el material es veu reduït a 0 euros gràcies al préstec del material, implicant una baixada de 104,28 euros del cost total.
- S'adquireix una direcció web amb un TLD gratuït (.tk), reduint el cost de 15,99 euros a 0.
- Degut a l'augment de les hores dedicades a la tasca T7, el consum d'electricitat i el sou del desenvolupador es veuen afectats. Calculant l'augment de 40 hores (equivalent a una setmana amb una jornada laboral de 8h), el consum d'electricitat augmenta 5,76kW (Secció 5.1), i el cost combinat ascendeix als 400,79 euros (0,79 euros en energia, 400 euros en sou).
- Els elements amb un cost mensual si que es veuen afectats degut a l'augment del temps de desenvolupament total. Afegint la desviació de 1 setmana a les 9 setmanes previstes a la Secció 4 i a conseqüència dels càlculs realitzats a la Secció 3.2, s'obté un cost temporal de 10 setmanes o 2 mesos i 2 setmanes. Per tant, no cal considerar un augment dels costos mensuals (Lloguer de local i pla d'internet) ja que no s'arriba a la següent fita mensual.

Concepte	Quantitat	Cost (Euros)		Amortització (Euros)	
		Unitari	Total	Anual	Mensual
Servidor	3	0	0	0	0
Material servidor	3	0	0	0	0
Enrutador	1	26,10	26,10	5,22	0,43
Connexió internet	3 (mesos)	18,15	54,45		
Domini web	1	0	0		
Ordinador	1	699,90	699,90	139,98	11,66
Electricitat	57,61 (kW)	0,135	7,78		
Enllumenat	2	10,95	21,90	2,19	0,18
Sou	380 (hores)	10	3800,00		
Local	3 (mesos)	420,00 mensual	1260,00		
Mobiliari	1	40	40	4	0,33
Total			5910,13	156,61	13,04

Taula 4: Relació actualitzada de materials i costos associats al projecte.

En conjunt, la desviació temporal comporta un augment de 280,51 euros sobre el pressupost inicial, situant-lo en un cost total de 5910,13 euros. La taula amb la relació de materials i els costos associats es pot trobar actualitzada a continuació. Les cel·les actualitzades respecte a la Taula 3 es troben marcades amb un color distintiu.

7 Especificació tècnica

7.1 Arquitectura

A nivell d'arquitectura les alternatives s'analitzen a la Secció 2. L'arquitectura triada és una arquitectura mestre-esclau (Secció 2.3); hi ha un servidor que actua com a gestor principal i els altres són servidors secundaris que s'encarreguen de l'accés a dades. Per tal d'evitar les rèpliques de dades cada cert temps i no crear problemes d'inconsistència entre els diferents servidors secundaris, es decideix usar una estratègia de transferència en temps real. Amb aquest mètode tots els servidors reben la informació dins d'un període de temps molt petit, evitant que hi hagi disparitats entre les versions de cada servidor.

Aquest mètode és difícil d'implementar, ja que cal tenir en compte la càrrega que suposa la transmissió constant de dades a la xarxa privada de l'arquitectura. Aprofitant que existeix una eina de codi lliure que gestiona aquest tipus de comunicació, s'ha decidit usar Apache ZooKeeper com a programari de gestió de dades.

7.1.1 Apache ZooKeeper

Apache ZooKeeper¹⁰, desenvolupat per *The Apache Software Foundation* des de l'any 2010, abstrau el procés de verificació de la integritat de les dades i de la configuració dels servidors. Aquesta eina de codi lliure desenvolupada en llenguatge Java permet comunicar diversos servidors i crear una única base de dades conjunta.

ZooKeeper s'encarrega de mantenir la integritat de les dades encara que hi hagi una desconexió per part d'uns quants servidors gràcies a la redundància que ofereix. Per tal de mantenir aquesta redundància, els creadors del programari recomanen que hi hagi un nombre senar d'elements dins de l'eixam o clúster de servidors. Això permet que, en cas de conflicte, no hi pugui haver un empat en el nombre de servidors amb les mateixes dades i es pugui determinar quin és el grup de servidors amb més prioritat. Una de les característiques de ZooKeeper és la seva velocitat per accedir a dades; aquestes s'emmagatzemen en memòria, aconseguint un throughput molt alt i mantenint la latència al mínim en cas de operacions de lectura. Per a les operacions d'escriptura la latència augmenta mínimament degut a la feina de replicació que comporta, però segueix sent molt menor que en un sistema convencional.

Aquest programari permet la modificació de l'eixam actiu gràcies a la seva configuració dinàmica. Quan s'inicia el programari per primer cop als servidors que conformen l'eixam, cada un d'ells llegeix la configuració estàtica d'un fitxer local i la guarda en memòria. Un cop llegida la configuració, intenta comunicar-se amb els servidors que hi ha indicats a la mateixa. Si es connecten dos o més servidors al mateix eixam, aquests comencen a realitzar l'intercanvi de dades amb un control de versions intern al programari. En pocs segons tots els servidors disposen de l'última versió disponible de les dades. Si un servidor amb les dades desactualitzades es connecta a un eixam en funcionament, se li dóna la qualitat d'*observador* de forma temporal. Els servidors marcats com a observadors no poden contribuir a la modificació de dades de l'eixam sense actualitzar les dades de les quals disposen; un cop actualitzades se'ls marca com a participants a l'eixam de servidors.

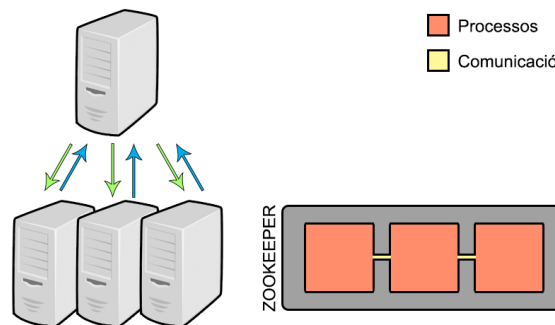


Figura 4: Arquitectura del projecte implementat amb els servidors esclau.

¹⁰zookeeper.apache.org

Gràcies a aquesta implementació, és possible augmentar o disminuir la quantitat de servidors de l'eixam en qualsevol moment: Cal informar a tots els participants, observadors i aspirants de l'eixam de quins servidors el formen. Les modificacions d'aquest tipus es guarden a la configuració en memòria, permetent realitzar canvis temporals de transferència o clonat de dades sense comprometre la integritat o la seguretat de l'eixam. Per tal de manipular les dades emmagatzemades dins els servidors, ZooKeeper ofereix una aplicació (client) que es connecta a un dels servidors de l'eixam mitjançant una direcció IP i un port.

7.1.1.1 Client d'Apache ZooKeeper

El client que ofereix Apache ZooKeeper és una aplicació senzilla en llenguatge Java que es connecta a un servidor local pertanyent a un eixam de ZooKeeper i permet interactuar amb les dades mitjançant una consola. Amb la consola es poden realitzar diferents operacions (Secció 7.1.4) que afecten a tot l'eixam, però cal configurar el sistema de registres prèviament. Aquesta restricció ve donada per el sistema d'estats d'Apache ZooKeeper: és imperatiu tenir informació sobre la connexió amb el servidor per evitar errors.

- Connectat - El client té accés complet al servidor i hi pot interactuar sense limitacions.
- Connectant - El client està pendent de rebre confirmació per a començar o continuar les operacions.
- Fallada d'autenticació - El client no s'ha pogut autenticar amb les credencials requerides.
- Desconnectat - El client no està connectat al servidor.

Les diferents transicions entre els estats del client es poden veure a la Figura 5, extreta de la documentació oficial de ZooKeeper.

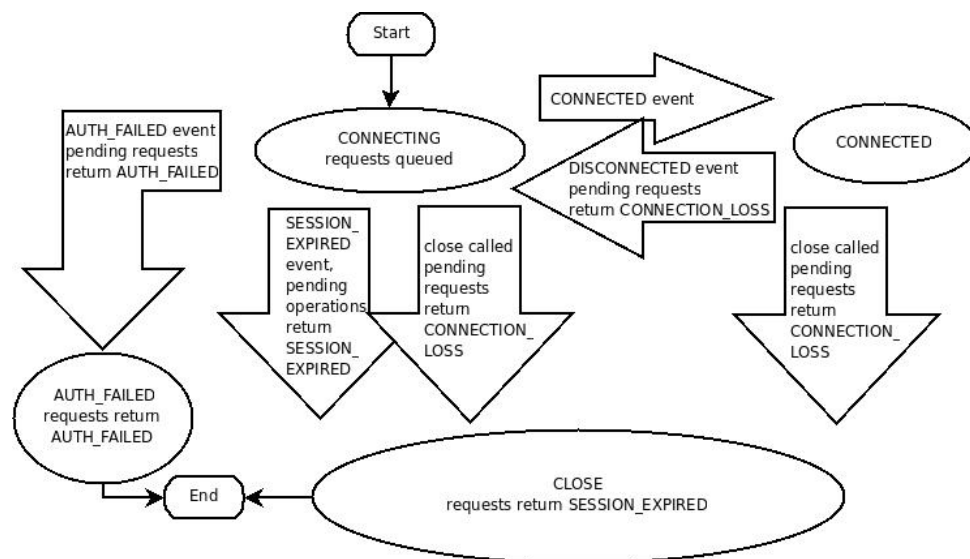


Figura 5: Transicions entre els estats de connexió d'un client d'Apache ZooKeeper.

7.1.2 Instal·lació del programari

La instal·lació i configuració d'Apache ZooKeeper no és senzilla. Per tal de facilitar i agilitzar la instal·lació del programari Apache ZooKeeper als servidors secundaris s'ha desenvolupat un *script* amb llenguatge Bash. Aquest script actualitza la llista de repositoris del servidor, descarrega els binaris per a la instal·lació de la versió 3.4.12 i configura les variables d'entorn necessàries per al funcionament del programari. El codi es pot veure a l'Annex C.1 amb una descripció del seu funcionament i ús.

7.1.3 Model de dades

ZooKeeper adopta un model de dades molt similar al de Unix, on les dades es guarden en directoris i cada directori pot contenir diversos fitxers o més directoris. Aquest model té el seu origen a una única carpeta i es desenvolupa en forma d'arbre. ZooKeeper, però, no distingeix entre directoris i fitxers; tots els elements presents al model de dades de ZooKeeper s'anomenen *zNodes*, i poden contenir informació (com un fitxer) i més *zNodes* al següent nivell de profunditat (com un directori). Per tal de mantenir la estructura, cada node guarda informació relativa a les dades d'usuari que emmagatzema i variables del sistema com la versió del node o la quantitat de subnodes que en depenen.

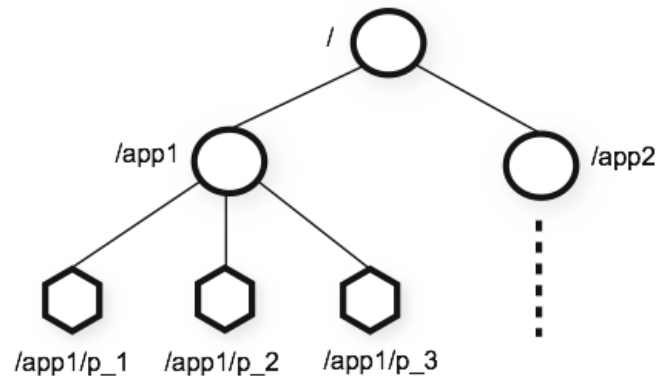


Figura 6: Exemple del sistema jeràrquic d'Apache ZooKeeper.

Tant el *zNode* arrel com la sintaxi de navegació són compartides amb els sistemes Unix: L'arrel té com a nom "/", i aquest mateix caràcter serveix com a separador per a accedir a subnodes. Per tant, `"/ruta/fins/fitxer"` és un fitxer vàlid, tal i com es pot veure a la Figura 6. Existeix, però, un node propi del sistema que manté la informació necessària per al seu correcte funcionament. Aquest node és `"/zookeeper/quota"`, i entre altres variables té l'espai ocupat per les dades d'Apache ZooKeeper.

De la mateixa forma que Unix, ZooKeeper ofereix la possibilitat de crear elements temporals. Els nodes *efímers* es troben lligats a una sessió de client, i un cop el client de ZooKeeper es desconnecta de l'eixam, els nodes efímers que ha creat s'eliminen automàticament.

7.1.4 Operacions

Un dels objectius d'aquest programari és mantenir una interfície de codi molt senzilla per tal que els desenvolupadors puguin usar-lo amb facilitat. Apache ZooKeeper ofereix 7 operacions de baix nivell diferents per a interactuar amb el sistema: Crear i esborrar un node, comprovar l'existència d'un node, emmagatzemar o llegir dades, obtenir els subnodes i esperar a la sincronització de tots els servidors de l'eixam. Segons la documentació que ofereix el programari, el sistema garanteix que aquestes operacions són atòmiques i per tant l'eixam no pot quedar en un estat inconsistent; o es completen correctament o es desfan els canvis. Amb la combinació d'aquestes operacions es pot realitzar un gran ventall d'operacions a més alt nivell, com la creació d'un node amb les dades emmagatzemades directament, o l'esborrat recursiu d'una estructura de nodes.

7.1.5 Garanties de funcionament

El sistema implementat per Apache ZooKeeper ofereix una sèrie de garanties crucials per al funcionament en temps real d'una aplicació que en depengui. Ja que un dels seus objectius és ser una base per aplicacions més complicades, com per exemple serveix que requereixin sincronització entre diferents sistemes, Apache ZooKeeper garanteix els següents punts:

- Les actualitzacions de dades sempre s'apliquen en l'ordre que s'envien.
- Les actualitzacions són atòmiques; o es completen correctament o es desfan els canvis intermedis.

- Un client que es connecti a l'eixam veurà les mateixes dades independentment de a quin servidor es connecti.
- Un cop s'ha aplicat una modificació, les dades persisteixen fins que una altra operació les modifica o esborra.
- Les dades que pot veure un client de ZooKeeper sempre són les dades més actualitzades (amb un cert marge de temps).

7.2 Aplicació web

Per tal de oferir una eina de visualització del correcte funcionament de l'arquitectura, s'ha decidit implementar una aplicació web que emmagatzema les dades dins dels servidors que conformen l'arquitectura. Aquesta aplicació web està oberta al públic per a que tothom la pugui usar. Per tal de garantir el seu funcionament, cal mantenir un servidor de peticions web, que s'encarrega de gestionar els accessos al domini adquirit per a la realització del projecte.

7.2.1 Apache

S'ha triat el servidor HTTP v2.4.25 d'Apache¹¹ degut a la seva qualitat de programari lliure i de codi obert. Aquest programari permet gestionar les peticions web realitzades a múltiples dominis dins d'un mateix servidor sense cap conflicte. Aquest servidor de peticions web revisa les capçaleres de les peticions per tal d'extreure el domini sol·licitat, i retorna com a resposta un fitxer HTML estàtic o actua com a *proxy* (intermediari) i redirigeix la petició cap a un altre programa.

La resposta amb un fitxer HTML sovint és per a informació estàtica, ja que Apache no és capaç de modificar el contingut de la resposta. Per tant, el projecte implementat usa el servidor HTTP d'Apache com a intermediari i redirigeix les peticions cap a un altre programari, Django, que s'encarrega d'omplir les pàgines web amb les dades actualitzades a temps real.

7.2.2 Django

Django és un programari que actua com a marc de treball per al servei de peticions web. Aquest programari permet modificar fitxers estàtics amb informació obtinguda de forma dinàmica. Tal i com s'indica a la Secció 2.2.3, existeixen molts marcs de treball similars; s'ha triat Django (v2.1.3) per el coneixement previ de l'autor amb el llenguatge de programació Python.

Aquest programari ofereix la possibilitat de cridar qualsevol codi en Python quan es rep una petició; consultar bases de dades, fer peticions web addicionals o llegir informació de fitxers del sistema són alguns dels exemples. Per tal d'extendre la seva funcionalitat, Django també ofereix un sistema de memòria cau (*cache*). Aquesta característica permet multiplicar el nombre de peticions per segon, ja que les pàgines web es poden emmagatzemar amb la informació adaptada i servir en un temps molt menor.

En aquest projecte Django s'encarrega de demanar les dades relatives a l'aplicació web, que es troben emmagatzemades als servidors secundaris. Un cop obtingudes les dades, aquestes es fan servir per omplir uns formularis estàtics que conformen cada una de les pàgines web de l'aplicació. Django és relativament segur per el que fa a atacs informàtics similars a XSS¹² i injeccions SQL. Tot i això, s'ha triat que tots els formularis estiguin restringits a caràcters alfanumèrics per tal que el projecte sigui el menys vulnerable possible.

¹¹httpd.apache.org

¹²Cross-site scripting, Viquipèdia. ca.wikipedia.org/wiki/Cross_Site_Scripting

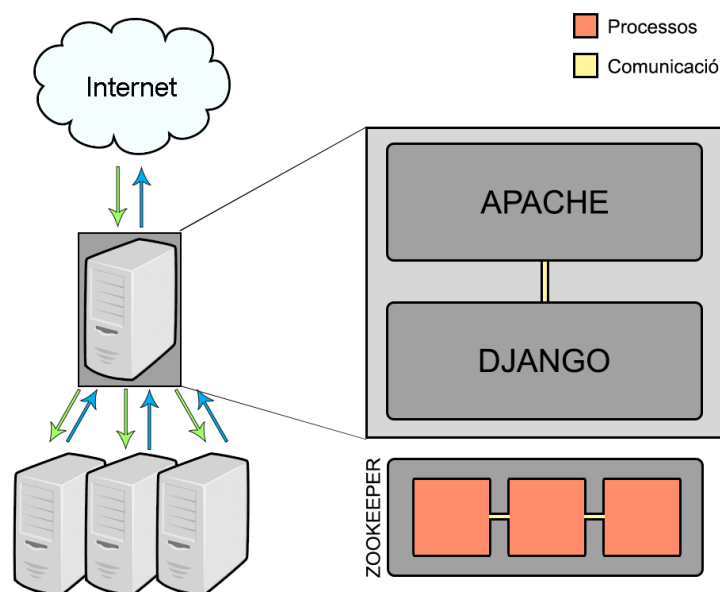


Figura 7: Arquitectura del projecte implementat amb l'aplicació web.

7.2.3 Funcionalitats

Degut als motius exposats a la Secció 1.1, l'aplicació web ofereix la possibilitat de gestionar les classificacions de tornejos amb múltiples participants. Per a gestionar aquests tornejos s'identifiquen els següents casos d'ús:

1. Crear un nou torneig.
2. Veure un llistat de tots els tornejos existents.
3. Consultar l'estat actual d'un torneig.
4. Actualitzar les posicions dels diferents participants dins el torneig.
5. Eliminar un torneig existent.

Per poder obtenir una referència de les operacions realitzades per a cada cas d'ús, es pot consultar l'Apèndix B.

7.2.3.1 Creació

Per evitar que els participants o usuaris aliens puguin editar les dades d'un torneig de forma il·lícita, s'ha decidit implementar un mot de pas quan es sol·licita una modificació. Aquest mot de pas o contrasenya és un camp requerit quan es vol crear un nou torneig (Veure Figura 8), i només el creador o gestor hi té accés. Aquest mot de pas s'emmagatzema als servidors i en cap cas s'envia la informació cap a l'aplicació web, evitant així el filtrat de dades. Els únics dos casos d'ús que requereixen l'enviament de dades dinàmiques són la consulta de classificació i la consulta de tornejos existents. En cas que les dades que es sol·liciten no estiguin disponibles, Django respon la petició web amb una aplicació d'error personalitzada que en detalla el motiu; els possibles causants d'aquests errors s'analitzen amb profunditat a la Secció 7.3.

7.2.3.2 Llistat

Tal i com es pot veure a la Figura 9, l'aplicació web que mostra els tornejos existents indica el nom de cada torneig i la quantitat de participants. Cada entrada té un enllaç que redirigeix a la classificació del propi torneig, i quan s'esborra un torneig existent aquest també desapareix del

Figura 8: Formulari de creació d'un torneig.

l·listat. En cas que no hi hagi cap torneig creat, s'ofereix a l'usuari un enllaç per a poder crear-ne un; hi ha un accés directe similar a la part superior de la pàgina, present en tot moment durant la navegació per l'aplicació web. Addicionalment s'ha afegit un camp de cerca per filtrar els tornejos segons el nom. En el moment que la llista de tornejos creix, es pot trobar el torneig buscat filtrant la llista per nom en lloc de desplaçar-se de forma manual per l'aplicació web.

Figura 9: Llistat de tornejos creats.

7.2.3.3 Consulta, edició i esborrat

Un cop creat un torneig, es permet editar la classificació dels participants que el formen. A la Figura 10 es pot veure el panell d'edició del torneig, accessible de forma oberta per a tots els usuaris. Aquest panell permet visualitzar els diferents nivells de la competició, on els nivells superiors representen enfrontaments més propers a la final, i el nivell més alt es troba reservat per al nom del guanyador. Tot i que la funcionalitat és la correcta, degut al desconeixement de la programació amb llenguatge web i manca de temps per al desenvolupament, el format del panell presentat no és fàcilment comprensible. Aquest detall s'ha anotat de cara a futures millores del projecte.

Per tal de modificar la classificació dins del panell, els usuaris poden prémer sobre els noms dels participants per marcar-los com a guanyadors de l'enfrontament. Aquesta funcionalitat és oberta a tothom, però els canvis no es veuen reflectits al sistema fins que s'omple el camp de "contrasenya" i es prem el botó d'"Actualitza" o "Esborra". Un cop es compleix aquest requisit, tant el mot de pas entrat per l'usuari com la codificació de la classificació s'envien al sistema. El sistema verifica que el mot de pas coincideix amb el que s'ha emmagatzemat al servidor i actualitza o esborra les dades.

7.2.4 Emmagatzematge de dades

Per tal d'obtenir un correcte funcionament de l'aplicació web només cal emmagatzemar dos tipus d'elements: tornejos i participants. Ja que un torneig consta de múltiples participants i els partici-

Partits de futbol
4 participants

Aztecs

Fusion

Aztecs

Badgers

Fusion

Friars

Aztecs

Figura 10: Panell de classificació d'un torneig.

pants són identificats per una cadena de text, s'ha decidit aprofitar l'estructura d'arbre que ofereix Apache ZooKeeper (Veure Secció 7.1.3): Hi haurà un node arrel que mantindrà la informació relativa al torneig i múltiples subnodes, un per a cada jugador, amb la informació respectiva. Tots els tornejos seran subnodes de `/tornejos`, per afavorir la reutilització de l'arquitectura en múltiples aplicacions. Apache ZooKeeper presenta una funcionalitat que permet la creació d'elements de forma automàtica sense que hi hagi col·lisions de noms. Aquesta funcionalitat és la creació de nodes "seqüencials"; aquests nodes tenen un sufix de 10 caràcters numèrics que actuen com a comptador, i el comptador és únic per a cada node.

La informació necessària per a definir l'estat d'un torneig i els seus jugadors consta de diferents elements, però els nodes d'Apache ZooKeeper només accepten que les dades d'usuari siguin una cadena de text (*string*). Per tal de encabir la informació necessària a cada node individual, s'ha decidit emmagatzemar les dades de cada node en format JSON i traduir-les en les operacions de lectura i escriptura.

7.2.4.1 Tornejos

Per aconseguir un funcionament mínim de l'aplicació web, cal emmagatzemar 3 variables: El nom del torneig, el mot de pas i la classificació dels jugadors. Per augmentar les funcionalitats s'ha decidit implementar dues variables més, el mode d'eliminació dels jugadors i la data d'esborrat del torneig. El tipus de cada variable es pot observar a la Taula 5, junt amb una petita descripció per a la funcionalitat que ofereix. Per tal de mantenir una organització amb els nodes dels tornejos creats, aquests usen l'identificador `t<id>`, on `id` és l'identificador usat per els nodes seqüencials de ZooKeeper. Per exemple, el node d'un torneig es pot referenciar amb la ruta `/tornejos/t0000000004`.

Variable	Tipus	Descripció
<code>name</code>	<code>char[32]</code>	Nom del torneig
<code>modality</code>	<code>int</code>	Mode d'eliminació del torneig (deshabilitat)
<code>password</code>	<code>char[16]</code>	Mot de pas del torneig
<code>classification</code>	<code>char[128]</code>	Classificació del torneig
<code>deletion_date</code>	<code>char[10]</code>	Data d'esborrat del torneig (dd/mm/aaaa)

Taula 5: Variables pròpies d'un torneig.

La variable `modality` no s'usa degut a la limitació que aporta l'aplicació web per a representar els diferents tipus de tornejos. Per exemple, un torneig basat en eliminació doble requereix més del doble de codi en HTML i JavaScript que un torneig amb eliminació simple. De cara al futur del projecte s'ha pensat en diferents formes d'implementar aquestes funcionalitats.

La variable `classification` indica la classificació del torneig i ha estat codificada segons els següents criteris:

- En un torneig amb N jugadors hi ha $N-1$ enfrontaments.
- Cada enfrontament té 3 possibles estats: No jugat (U), Participant 1 guanya (1), Participant 2 guanya (2).

- A cada ronda hi ha $M/2$ enfrontaments, on M és el nombre de participants de la ronda.
- A cada ronda hi ha la meitat de participants que a la ronda anterior.
- Per determinar els participants d'una ronda cal saber quins participants han guanyat a la ronda anterior

Per tant, en un torneig amb N jugadors, la variable de classificació tindrà $N-1$ caràcters entre "U", "1" i "2", segons els resultats dels enfrontaments. Els primers $N/2$ caràcters corresponen a la primera ronda, els $N/4$ següents a la segona ronda, i així successivament fins arribar a la ronda final, equivalent a l'últim caràcter. Per simplicitat a l'hora de representar els tornejos a l'aplicació web s'ha decidit que la quantitat de participants sigui una potència de 2; aquesta restricció també assegura que la variable de classificació pugui ser més senzilla de codificar.

Finalment, la variable `deletion_date` és un indicador usat per eliminar el torneig de forma automàtica. Cada 24h s'activa un *script* que es connecta a la base de dades i comprova que tots els tornejos encara tinguin vigència; per defecte, aquesta variable té com a valor la data de 30 dies després de la creació del torneig.

7.2.4.2 Participants

De cara al futur del projecte s'han implementat diverses variables que actualment no s'usen, però que són necessàries per a un funcionament total de l'aplicació. Totes les variables definides es poden veure a la Taula 6. De forma similar als tornejos, els participants s'emmagatzemen en nodes amb l'identificador `p<id>` i com a subnode del torneig propi. Per exemple, el node d'un jugador es pot referenciar amb la ruta `/tornejos/t0000000004/p0000000001`.

Variable	Tipus	Descripció
name	char[32]	Nom del participant
disqualified	int	Determina si el participant pot seguir tenint enfrontaments
points	int	Punts obtinguts per el participant
wins	int	Enfrontaments guanyats
losses	int	Enfrontaments perduts

Taula 6: Variables pròpies d'un participant.

És precís fer una distinció entre les variables `points` i `wins`: En certes modalitats una victòria comporta el guany de més d'un punt, com és el cas de les lligues de futbol o les competicions de carreres. Ja que el guanyador es calcula a partir d'una fórmula basada en els punts i no les victòries, es requereix aquest camp addicional.

7.3 Comunicacions

Un cop establerts els programaris que s'usaran per a l'arquitectura i l'aplicació web, cal definir de quina forma es comunicaran. Ja que el programari Zookeeper està desenvolupat en el llenguatge de programació Java, cal poder comunicar-lo amb els processos de Django, implementat amb llenguatge Python. Kazoo¹³ és una llibreria de Python que permet realitzar aquesta comunicació i actua com a embolcall per a les crides de Zookeeper. Aquesta llibreria permet crear un client que es comunica amb el conjunt de servidors gestionats per diversos processos de Zookeeper, interactuar amb ells emmagatzemant i accedint a dades, i modificar la configuració dels servidors sense haver de reiniciar-los. Per a tal de treballar amb la versió d'Apache ZooKeeper 3.4.12, cal instal·lar la versió de Kazoo 2.6.0.

Degut al mètode de funcionament d'Apache i Django, on s'instancia un fil per a cada petició entrant, és necessari valorar si el client de Kazoo es vol instanciar per a cada petició o si es vol mantenir un sol client encès de forma persistent.

- Si s'instancia un client per a cada petició entrant, cal afegir el temps de connexió al conjunt de servidors al temps total de resposta.
- Si es manté el client de forma permanent, cal establir un mètode de comunicació amb aquest, ja que Django no facilita l'existència d'objectes persistents dins del procés.

¹³Kazoo. Desenvolupat per Nimbus Project, 2011, mantingut per la comunitat. kazoo.readthedocs.io

Valorant les dues alternatives, s'arriba a la conclusió que la segona opció resulta més escal·lable. En un sistema on poden arribar a entrar milers de peticions per segon no és viable sobrecarregar encara més el temps de resposta mitjançant connexions intermèdies. Llavors, cal identificar la millor opció per a la comunicació amb el client de Kazoo. Paral·lelament, sorgeix el problema de l'escal·labilitat amb un únic client: És capaç de gestionar milers de peticions? En conseqüència es decideix implementar el mateix model amb múltiples clients de Kazoo.

La gestió de la comunicació entre els clients i els servidors es simplifica en una API que actua de pont entre cada client de Kazoo i el fil de Django pertinent. La API indicada corre com a un procés de Python dins del servidor de front-end, i és l'encarregada de gestionar les peticions que realitza l'aplicació web cap al client de Kazoo. Per aquest projecte s'ha decidit instanciar un procés d'API per a cada client de Kazoo i assignar el procés a un port del servidor local, començant per el port 6000 i de forma consecutiva. La configuració necessària per a establir els servidors i els ports disponibles es troba emmagatzemada en un fitxer accessible per tots els processos involucrats. En el moment que un fil de Django necessita realitzar una operació amb les dades de ZooKeeper, s'instancia una connexió amb l'API mitjançant una selecció aleatòria dels port disponibles. Tota la comunicació amb la API es realitza mitjançant el mòdul de Python *multiprocessing*¹⁴, que permet la transmissió de variables i classes de Python sense cap operació addicional per part dels usuaris. Amb aquest mòdul es poden transmetre variables de tipus diccionari; el format usat per a la petició entre el procés de Django i l'API es basa en un identificador de text sota el nom **operation** i un subdiccionari amb les dades necessàries sota el nom **data**. Les respostes que envia l'API cap al fil de Django són un codi de resposta, que indica si la operació s'ha completat correctament, i les dades associades a la operació. Per exemple, per a la creació d'un torneig s'intercanvien els missatges de la Taula 7. El codi necessari per mantenir la comunicació amb el procés de l'API es pot consultar a l'Annex C.3, tant per la part del client com per la part del servidor.

Petició	Resposta
<pre>{ 'operation' : 'create', 'data' : { 'name' : 'Torneig exemple', 'modality' : 0, 'password' : 'Exemple', 'players' : ['P1', 'P2', 'P3', 'P4'] } }</pre>	<pre>{ 'code' : 0, 'data' : '/tornejos/ t00000000004' }</pre>

Taula 7: Variables transmeses per a la petició i resposta de la creació d'un torneig amb l'API de comunicació.

A la Figura 11 es pot veure l'arquitectura final del projecte implementat de forma física i amb un esquema de les comunicacions i els processos involucrats. Tal i com s'indica a la llegenda, cada fil de processat disposa dels mateixos recursos que qualsevol altre fil del mateix procés; és a dir, si un fil de Django es pot comunicar amb tots els processos de l'API i en conseqüència amb tots els processos de Kazoo, tots els fils de Django disposen d'aquesta capacitat.

7.3.1 Connectivitat amb ZooKeeper

El procés d'API manté una instància de Kazoo connectada en tot moment, i en cas de fallada del procés aquest es torna a iniciar automàticament. Si el client de Kazoo identifica que no és possible connectar amb els servidors d'Apache ZooKeeper, retornarà un codi d'error i la comunicació es tornarà a intentar amb un altre port. En cas que cap dels servidors estigui disponible, s'informarà a l'usuari que ha enviat la petició via web que no és possible accedir a les dades.. Amb l'objectiu

¹⁴docs.python.org/3.5/library/multiprocessing.html

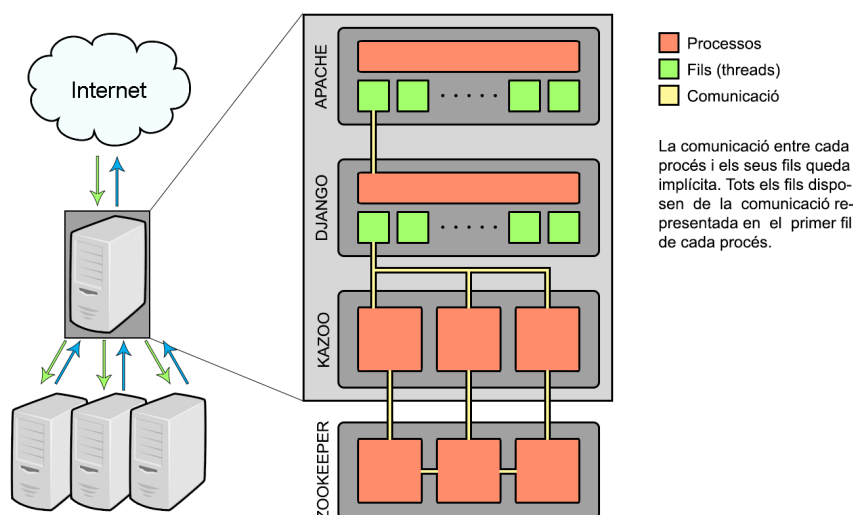


Figura 11: Esquema de referència del projecte desenvolupat.

de mostrar aquests missatges per mitjà de l'aplicació web, s'ha desenvolupat una vista especial que es pot mostrar des de qualsevol direcció web (Veure Figura 12).

S'ha trobat un error

No nodes available (Codi -1)

Figura 12: Aplicació web mostrant un missatge d'error.

7.3.2 Condicions de carrera

Com que el projecte s'ha desenvolupat en un entorn amb múltiples processos i fils, és possible que apareguin condicions de carrera. Aquest terme indica l'accés simultani de dos o més processos a un mateix recurs, en aquest cas un node de ZooKeeper, i és un problema que cal solucionar abans que s'introdueixin inconsistències en el sistema. Per evitar aquestes condicions, s'ha aprofitat que Kazoo ofereix un sistema de cadenats¹⁵ sobre els nodes de ZooKeeper que permeten exclusions mútues entre els diferents processos concurrents. Kazoo implementa el sistema de cadenats mitjançant la creació de nodes efímers (Secció 7.1.3) com a subnodes del node a bloquejar. Per tant, si el client de Kazoo té cap problema i acaba desconnectant-se de forma imprevista, tots els cadenats adquirits s'alliberen immediatament.

Si, per exemple, un usuari A demana les dades d'un torneig a l'aplicació web mentre un altre usuari B les ha editat, pot ser que les dades de l'usuari A tinguin inconsistències tot i l'atomicitat de ZooKeeper. Les inconsistències s'introdueixen al treballar amb diversos nodes: És possible que el node del torneig ja s'hagi actualitzat però els nodes dels participants encara tinguin les dades antigues. Repetint el mateix escenari amb cadenats es pot comprovar la seva efectivitat: L'usuari A aplica un cadenat d'escriptura sobre el node del torneig i comença a actualitzar les dades. L'usuari B intenta accedir a les dades però el sistema detecta que hi ha un cadenat i espera a que s'alliberi. Un cop l'usuari A ha acabat d'actualitzar les dades, allibera el cadenat del torneig i permet l'accés de tots els altres processos a la informació, ja actualitzada.

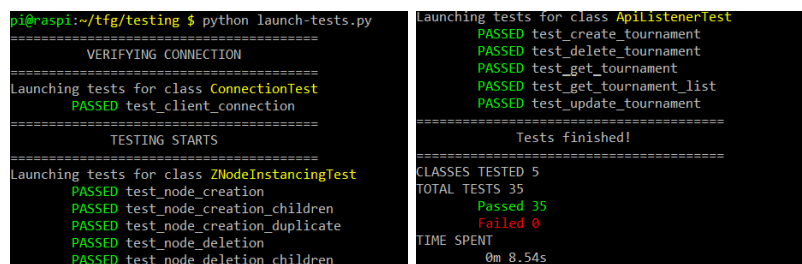
¹⁵ [ca.wikipedia.org/wiki/Cadenat_\(informàtica\)](https://ca.wikipedia.org/wiki/Cadenat_(informàtica))

8 Anàlisi del projecte

Un aspecte crucial per al desenvolupament d'un projecte és la validació del seu funcionament. En aquesta secció s'analitza el funcionament, rendiment i escal·labilitat del projecte desenvolupat, tant en la seva arquitectura com per l'aplicació web, per mitjà de diferents tipus de validacions.

8.1 Proves manuals i automatitzades

Per tal de validar el correcte funcionament de tot el sistema s'han desenvolupat diferents proves automàtiques a nivell de codi, basades en un marc de treball de *unit testing*. Degut al mètode de treball de desenvolupament basat en proves, era crucial desenvolupar primer les proves automatitzades i verificar que les funcionalitats fallaven. Després d'implementar el codi s'ha verificat que les proves anteriors ara compleixen els requisits. Ja que gairebé la totalitat del projecte ha estat desenvolupat en llenguatge Python, s'ha decidit implementar les proves amb el mateix llenguatge per tal d'accelerar la validació.



```
pi@raspi:~/tfg/testing $ python launch-tests.py
=====
VERIFYING CONNECTION
=====
Launching tests for class ConnectionTest
PASSED test_client_connection
=====
TESTING STARTS
=====
Launching tests for class ZNodeInstantiatingTest
PASSED test_node_creation
PASSED test_node_creation_children
PASSED test_node_creation_duplicate
PASSED test_node_deletion
PASSED test_node_deletion_children

Launching tests for class ApiListenerTest
PASSED test_create_tournament
PASSED test_delete_tournament
PASSED test_get_tournament
PASSED test_get_tournament_list
PASSED test_update_tournament
=====
Tests finished!
=====
CLASSES TESTED 5
TOTAL TESTS 35
Passed 35
Failed 0
TIME SPENT
0m 8.54s
```

Figura 13: Extractes de la sortida del *script* de proves.

En lloc d'usar el mòdul *unittest*¹⁶, s'ha implementat un *script* que actua de la mateixa forma i ofereix més personalització per a la validació de les proves. Els resultats de les proves s'organitzen per tipus de prova i s'indica si cada prova individual s'ha realitzat satisfactòriament, tal i com es pot veure a la Figura 13). En cas que una de les proves no tingui el resultat esperat, s'indica que ha fallat i s'adjunta un missatge d'error descriptiu.

Servidor	Port	OK	Missatge
127.0.0.1	6000		CONNECTED
192.168.1.15	6001		CONNECTED
192.168.1.16	6002		CONNECTED

Figura 14: Taula amb la informació de l'estat dels diferents servidors.

En cas que hi hagi cap problema amb l'accés a les dades, és important poder saber si els servidors funcionen correctament. S'ha desenvolupat una aplicació web addicional que sol·licita l'estat dels servidors als clients de Kazoo. Els clients responen amb un codi d'estat i un missatge rellevant. S'itera per tots els clients configurats i s'afegeixen les respostes de cada client a una taula HTML (Veure Figura 14). Amb una sola consulta és possible saber si tots els servidors funcionen correctament o si s'han de realitzar accions rectificatòries sobre cap d'ells, com es pot veure a l'exemple de la Figura 15.

Servidor	Port	OK	Missatge
127.0.0.1	6000		CONNECTED
192.168.1.15	6001		Server unavailable
192.168.1.16	6002		CONNECTED

Figura 15: Taula amb la informació de l'estat dels diferents servidors, amb un error.

Seguint la planificació establerta i concretament la Tasca T10, s'ha demanat a diferents voluntaris que analitzin el funcionament de l'aplicació web per trobar errades de desenvolupament. S'han definit una sèrie de tasques que els voluntaris han de realitzar en l'ordre establert, i posteriorment han de reportar si hi ha cap missatge d'error o comportament no esperat. Finalment, s'ha ofert un temps acotat per a que els voluntaris investiguin lliurement i reportin la facilitat d'ús de

¹⁶docs.python.org/3.5/library/unittest.html

la aplicació web i comentaris opcionals. Les respostes anònimes dels voluntaris han sigut altament satisfactòries ja que no s'ha trobat cap errada a l'aplicació web i en general es considera que és fàcil d'usar; només s'han indicat recomanacions per a la part visual.

8.2 Tolerància a fallades

Un dels avantatges que ofereix ZooKeeper respecte a la instal·lació de diversos servidors replicats manualment és la tolerància a fallades. La redundància que ofereix ZooKeeper per mitjà de programari és un factor crucial per mantenir un sistema consolidat i sense *downtime* o temps d'indisponibilitat; aquesta propietat permet que l'eixam segueixi funcionant fins i tot quan hi ha servidors inestables o amb problemes de connectivitat amb els altres servidors.

```

Displaying stored zNodes in tree structure (127.0.0.1)

/tornejos
/tornejos/t0000000002 >>> "b":{"deletion_date": "18/02/2019", "password": "asdf", "classification": "2121U12U11U02U0", "modality": 0, "name": "E Sports"}"
/tornejos/t0000000002/p0000000010 >>> "b":{"losses": 0, "wins": 0, "name": "CDE", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000009 >>> "b":{"losses": 0, "wins": 0, "name": "G1", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000008 >>> "b":{"losses": 0, "wins": 0, "name": "OIG", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000007 >>> "b":{"losses": 0, "wins": 0, "name": "FNC", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000006 >>> "b":{"losses": 0, "wins": 0, "name": "ASD", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000005 >>> "b":{"losses": 0, "wins": 0, "name": "ZAC", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000004 >>> "b":{"losses": 0, "wins": 0, "name": "BGM", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000015 >>> "b":{"losses": 0, "wins": 0, "name": "RTV", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000003 >>> "b":{"losses": 0, "wins": 0, "name": "FGH", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000014 >>> "b":{"losses": 0, "wins": 0, "name": "JKL", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000002 >>> "b":{"losses": 0, "wins": 0, "name": "UIO", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000013 >>> "b":{"losses": 0, "wins": 0, "name": "SKT", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000001 >>> "b":{"losses": 0, "wins": 0, "name": "VBN", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000012 >>> "b":{"losses": 0, "wins": 0, "name": "QWE", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000000 >>> "b":{"losses": 0, "wins": 0, "name": "ABC", "disqualified": 0, "points": 0}"
/tornejos/t0000000002/p0000000011 >>> "b":{"losses": 0, "wins": 0, "name": "SDJ", "disqualified": 0, "points": 0}"
/tornejos/t0000000000 >>> "b":{"deletion_date": "18/02/2019", "password": "asdf", "classification": "UUU", "name": "Partits de futbol", "modality": 0}"
/tornejos/t0000000000/p0000000003 >>> "b":{"losses": 0, "wins": 0, "name": "Badgers", "disqualified": 0, "points": 0}"
/tornejos/t0000000000/p0000000002 >>> "b":{"losses": 0, "wins": 0, "name": "Fusion", "disqualified": 0, "points": 0}"
/tornejos/t0000000000/p0000000001 >>> "b":{"losses": 0, "wins": 0, "name": "Friars", "disqualified": 0, "points": 0}"
/tornejos/t0000000000/p0000000000 >>> "b":{"losses": 0, "wins": 0, "name": "Aztecs", "disqualified": 0, "points": 0}"
/tornejos/t0000000001 >>> "b":{"password": "asdf", "classification": "2", "name": "Billar", "modality": 0, "deletion_date": "18/02/2019"}"
/tornejos/t0000000001/p0000000001 >>> "b":{"losses": 0, "wins": 0, "name": "Murray", "disqualified": 0, "points": 0}"
/tornejos/t0000000001/p0000000000 >>> "b":{"losses": 0, "wins": 0, "name": "Johnson", "disqualified": 0, "points": 0}"

/zookeeper
/zookeeper/quota

```

Figura 16: Bolcat de dades d'un servidor amb Apache ZooKeeper.

Tal i com s'indica a la Secció 7.1.1, en cas que un dels servidors es desconnecti de l'eixam de forma temporal i es torni a connectar passat uns instants, Apache ZooKeeper s'encarrega d'actualitzar les dades amb el seu control de versions intern. Així doncs, també és consistent amb les errades que pugui tenir un servidor, ignorant les dades inconsistents i aplicant la versió més nova per a tot l'eixam. Per a poder validar que les dades es propaguen correctament, s'ha desenvolupat un *script* que es connecta a un servidor donada la seva IP i escriu a la consola els continguts del servidor indicat (Veure Figura 16). El codi i l'explicació del seu funcionament es pot trobar a l'Annex C.2.

Registrant les dades disponibles amb tots els servidors en funcionament, s'ha realitzat una prova manual per a verificar que la propagació es realitza de forma correcta.

1. Aprofitant el script indicat, es realitza un bolcat de les dades de tots els servidors i es verifica que siguin idèntiques.
2. Es desconnecta un dels servidors de l'eixam, aturant el procés d'Apache ZooKeeper.
3. Es realitzen modificacions a les dades mitjançant l'aplicació web.
4. Es verifica que les dades s'han replicat correctament als servidors que pertanyen a l'eixam.
5. S'inicia el procés d'Apache ZooKeeper al servidor desconnectat anteriorment.
6. Passats uns instants per a que la connexió es realitzi correctament, es torna a usar el script per observar les dades del servidor reconnectat.
7. Es verifica que les dades s'han propagat correctament fins al servidor, i que les dades són idèntiques a tots els servidors.

8.3 Rendiment

Per avaluar el rendiment de l'aplicació web i poder estimar la càrrega que suporta el sistema, s'han realitzat una sèrie de proves usant Apache JMeter¹⁷. JMeter és una eina que permet analitzar el

¹⁷jmeter.apache.org

temps de resposta d'aplicacions web i recursos en línia. Aquest programari de codi lliure permet fer una gravació amb diferents accessos a recursos en línia per després replicar-los simulant més peticions per segon. Un cop s'inicia el procés de proves, les dades es poden recollir en temps real en gràfiques i estadístiques. Això habilita la realització d'un anàlisi al cap de pocs segons d'iniciar les proves.

La primera prova a realitzar consisteix en analitzar el funcionament de l'aplicació a plena capacitat per obtenir un marc de referència. Les dades recollides són el temps de resposta mitjà, la desviació en el temps de resposta i el *throughput* total, i permeten realitzar una comparació amb diferents casos posteriorment. Aquests tres tipus de dades es poden identificar a les següents figures per el seu color blau, vermell i verd respectivament.

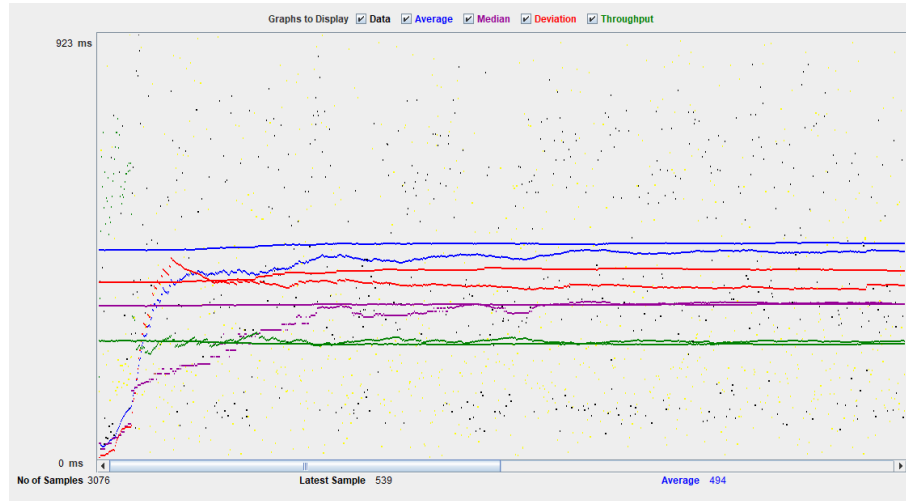


Figura 17: Temps de resposta de l'aplicació web.

Per a determinar el cas base es generen 10 clients ficticis que repetiran una sèrie d'operacions durant 60 segons. Això permet obtenir una relació senzilla entre les dades i el temps transcorregut. Les dades obtingudes a la prova base (Figura 17) demostren un rendiment acceptable de l'aplicació: Una mitjana de 494 mil·lisegons de resposta, amb una desviació de 420 mil·lisegons i un *throughput* total de 1.519,54MB (equivalent a 25,33 MB/s). Amb aquestes dades es pot confirmar que amb un funcionament nominal del projecte, l'aplicació web respon a l'objectiu de temps de resposta inferior als 3 segons.

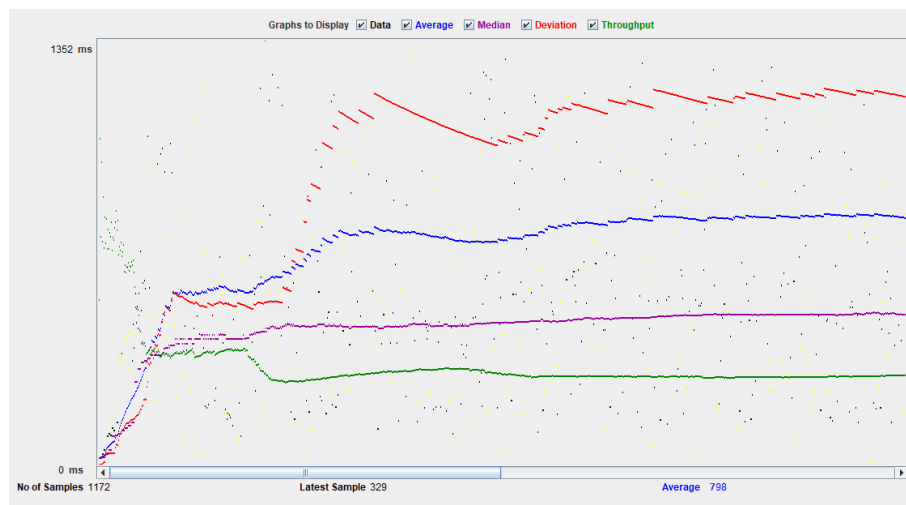


Figura 18: Temps de resposta de l'aplicació web, amb un servidor afectat.

Un cas interessant és el comportament del sistema en el cas de fallada d'un dels servidors. A la Figura 18 es poden observar les variacions que introdueix la desconexió manual d'un servidor. Es pot veure com el valor de *throughput* cau immediatament i tant el temps mitjà de resposta

com la desviació d'aquest augmenten considerablement. El temps de resposta mitjà s'incrementa en 300 mil·lisegons i la desviació del mateix augmenta gairebé 700 mil·lisegons. Tot i això, es pot comprovar que el sistema segueix funcionant correctament malgrat tenir un temps de resposta superior.

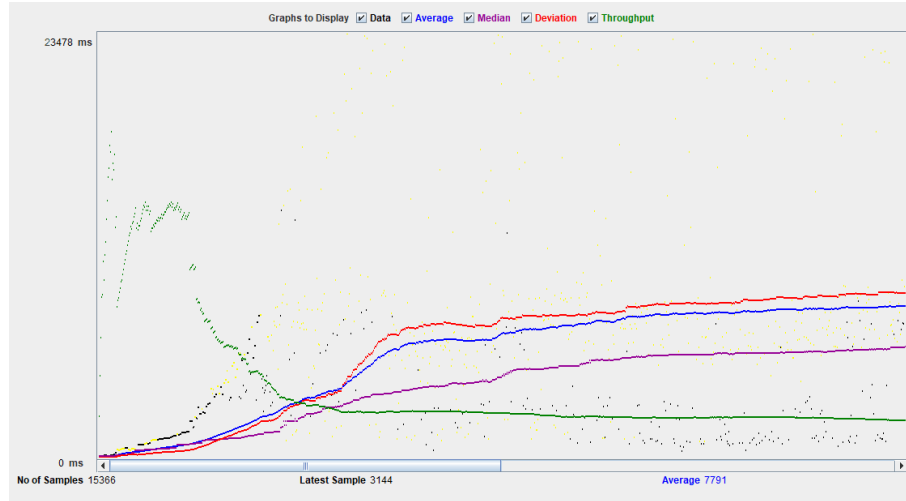


Figura 19: Temps de resposta de l'aplicació web amb sobrecàrrega.

Finalment, s'ha volgut sobrecarregar el sistema per veure quina capacitat pot mantenir (Figura 19). S'ha elevat el nombre de clients a 200 i la freqüència de peticions per segon de cada client s'ha doblat; els resultats són els esperats. El sistema comença a respondre peticions amb un rendiment molt superior al mostrat a la prova base ja que no havia assolit el límit de peticions, però ràpidament es comença a saturar. El throughput cau a una quarta part de l'original i el temps de resposta de l'aplicació augmenta gradualment fins als 8 segons, incloent una desviació de 7 segons addicionals. Gràcies a JMeter es pot comprovar com algunes de les peticions no aconsegueixen obtenir resposta, indicant la fallada del temps de servei de 100% del sistema. Malgrat tot, el sistema aconsegueix mantenir una taxa de resposta del 97.3%, també complint l'objectiu d'uptime superior al 95%.

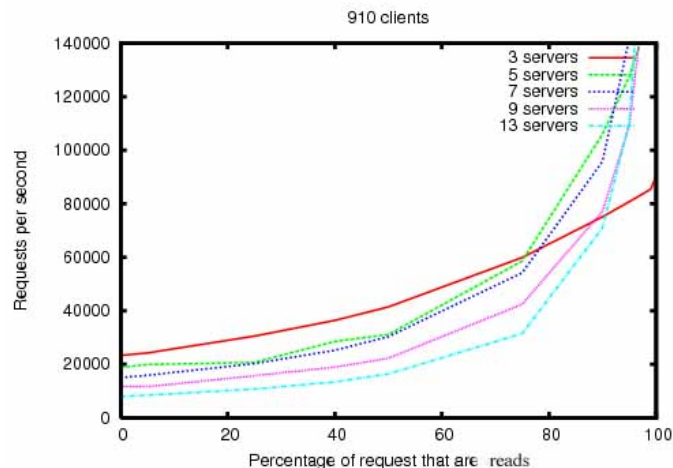


Figura 20: Relació entre ratio lectura-escriptura i peticions realitzades, segons dades oficials d'Apache ZooKeeper.

Segons la informació disponible a la plana web d'Apache ZooKeeper, un eixam amb 3 servidors és capaç de suportar una càrrega d'entre 20.000 i 140.000 peticions per segon depenent del percentatge d'operacions de lectura i escriptura (Veure Figura 20). Els servidors usats en el projecte són Raspberry Pi Zero W, que consten de 512MB de memòria RAM i un únic core de 1GHz; també cal tenir en compte que hi ha moltes capes intermèdies que afegeixen petits retards a totes les peticions. Tenint en compte que els servidors als quals tenen accés les grans empreses disposen

de molts més recursos, es pot obtenir una idea del rendiment que podria arribar a adquirir el sistema en un cas a gran escala. Un altre factor per a la diferència entre els rendiments de les dades oficials i el projecte implementat és el coll d'ampolla que presenta l'arquitectura implementada. És possible que part del problema de rendiment vingui donat per Apache o Django, ja que només tenen un procés que rep les peticions inicialment. Per tal de poder eliminar aquest factor, es podria implementar un programari de balanceig de càrrega amb diferents servidors que actuïn com a front-end.

9 Conclusions

El problema plantejat inicialment en aquest projecte és el problema d'escal·labilitat que es presenten certes architectures usades per grans empreses. Es realitza un anàlisi de les possibles alternatives, tant per architectures com per programaris, i s'implementa una solució basada en Apache ZooKeeper. Per tal de validar el correcte funcionament de l'arquitectura, es desenvolupa una aplicació web basada en Apache i Django, que emmagatzemen les dades amb el programari de ZooKeeper. L'aplicació web ofereix la capacitat de crear i gestionar competicions o tornejos, permetent l'accés al públic i alhora assegurant la privacitat de les dades. Un cop desenvolupada la solució, s'integren diferents proves automàtiques i manuals que permeten estimar el rendiment del sistema; els resultats són altament satisfactoris, tant per el rendiment de l'aplicació com per la satisfacció dels usuaris que ajuden amb la realització de les proves manuals.

El codi del projecte desenvolupat es pot trobar al repositori github.com/fndh/tfg.
L'aplicació web es pot visitar seguint l'enllaç tornejos.tk.

Pla de futur

S'ha previst continuar amb el desenvolupament del projecte per tal de cobrir més tipus diferents d'enfrontaments per l'aplicació web que no s'han pogut implementar degut a les limitacions temporals del projecte. Amb la col·laboració d'experts en l'àmbit del desenvolupament web es pot aconseguir una aplicació web que ofereixi les opcions necessàries per a la gestió de competicions de forma gratuïta i sense requerir dades personals, característiques que augmentarien la popularitat de l'aplicació web de forma exponencial.

Per evitar la problemàtica del coll d'ampolla a l'aplicació web, s'implementarà un sistema de balanceig de càrrega. Aquest sistema reparteix de forma equitativa les peticions que rep l'aplicació entre diferents servidors, de forma semblant a l'arquitectura mestre-esclau però amb els servidors de front-end. Aquesta implementació augmentarà la quantitat de peticions que es poden respondre per segon, i en cas de fallada d'un dels servidors de front-end l'afectació serà molt menor o fins i tot nul·la.

Agraïments

Vull agrair el suport de totes les persones que m'han acompanyat durant la realització del Treball Final de Grau, especialment al tutor del projecte, Jordi Guitart, per la seva orientació constant i ajuda tècnica. A en Kilian i a Cristina per el préstec dels servidors, i a en Fran per l'ajuda amb el desenvolupament de l'aplicació web.

També agraeixo l'ajuda presentada per els companys de l'associació SVGA al Campus Nord, per la idea de l'aplicació web i per la lògica requerida per la gestió dels tornejos.

Finalment, vull agrair especialment la col·laboració dels companys i amics que m'han ajudat durant el cursat del Grau en Enginyeria Informàtica.

Bibliografia

- Cardellini, V., & Casalicchio, E. (2002). The state of the art in locally distributed web-server systems. *Computer Science*, 265-266.
- Henderson, C. (2006). *Building scalable web sites*. O'Reilly Media Inc.
- Menascé, D. (2002, Nov). Qos issues in web services. *IEEE Internet Computing*, 6(6). doi: 10.1109/MIC.2002.1067740
- Netcraft. (2018). *September 2018 web server survey*. news.netcraft.com/archives/2018/09/24/september-2018-web-server-survey.html. ([Internet; descarregat Setembre 2018])
- Özsu, M., & Valduriez, P. (2011). *Principles of distributed database systems*. Springer New York. Retrieved from <https://books.google.es/books?id=TOBaLQMuNV4C>
- Plekhanova, J. (2009). Evaluating web development frameworks: Django, ruby on rails and cakephp. *Institute for Business and Information Technology*.
- Stats, I. W. (2018). *World internet users and 2018 population stats*. www.internetworldstats.com/stats.htm. ([Internet; descarregat Setembre 2018])
- Tanenbaum, A. S., & Van Steen, M. (2007). *Distributed systems: principles and paradigms*. Prentice-Hall.
- Ølnes, S., Ubacht, A., & Janssen, M. (2017). Blockchain in government: Benefits and implications of distributed ledger technology for information sharing. *Government Information Quarterly*, 34(3). Retrieved from <http://www.sciencedirect.com/science/article/pii/S0740624X17303155>

Annexos

A Esquemes de Gantt

Els esquemes de Gantt representats inclouen els mesos de desenvolupament del projecte, i els mesos es troben subdividits en les setmanes que els comprenen. Cada divisió vertical indica a quina setmana correspon mitjançant l'anotació del primer dia de setmana a la casella corresponent. Cada tasca principal està identificada per les línies horitzontals de colors, i les seves subtasques estan indicades als blocs inferiors, en color gris dins l'esquema temporal.

Les tasques es troben classificades per colors segons la seva criticalitat per al projecte. Les tasques que suposen una desviació major per la planificació en el cas que no es puguin acabar amb el temps previst tenen un color més roig, i les que suposen una desviació menor es troben indicades amb un color més verd. Les dependències entre les diferents tasques es poden identificar mitjançant les fletxes que comencen al bloc de finalització d'una tasca i arriben fins el bloc l'inici d'una altra tasca.

Planificació inicial

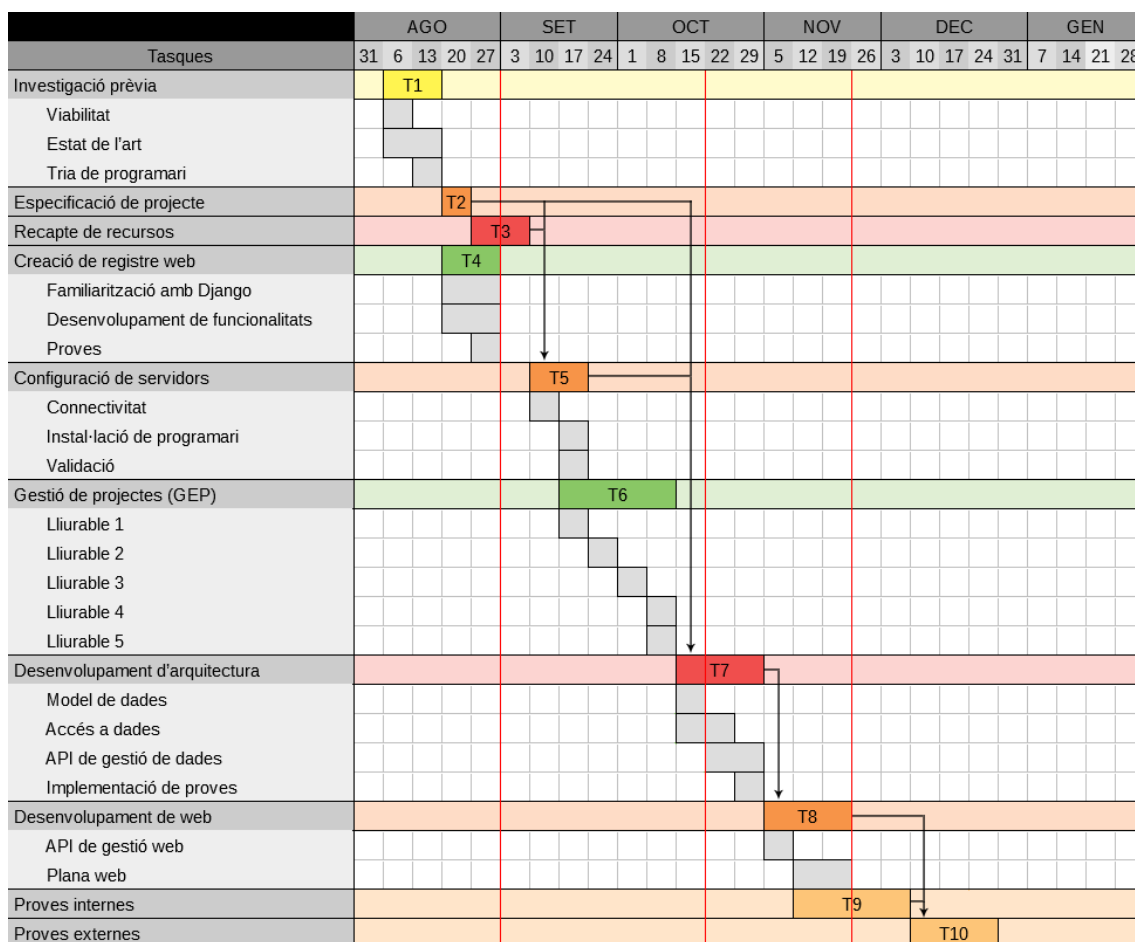


Figura 21: Esquema de Gantt representatiu de la planificació inicial.

Planificació ajustada a les desviacions temporals

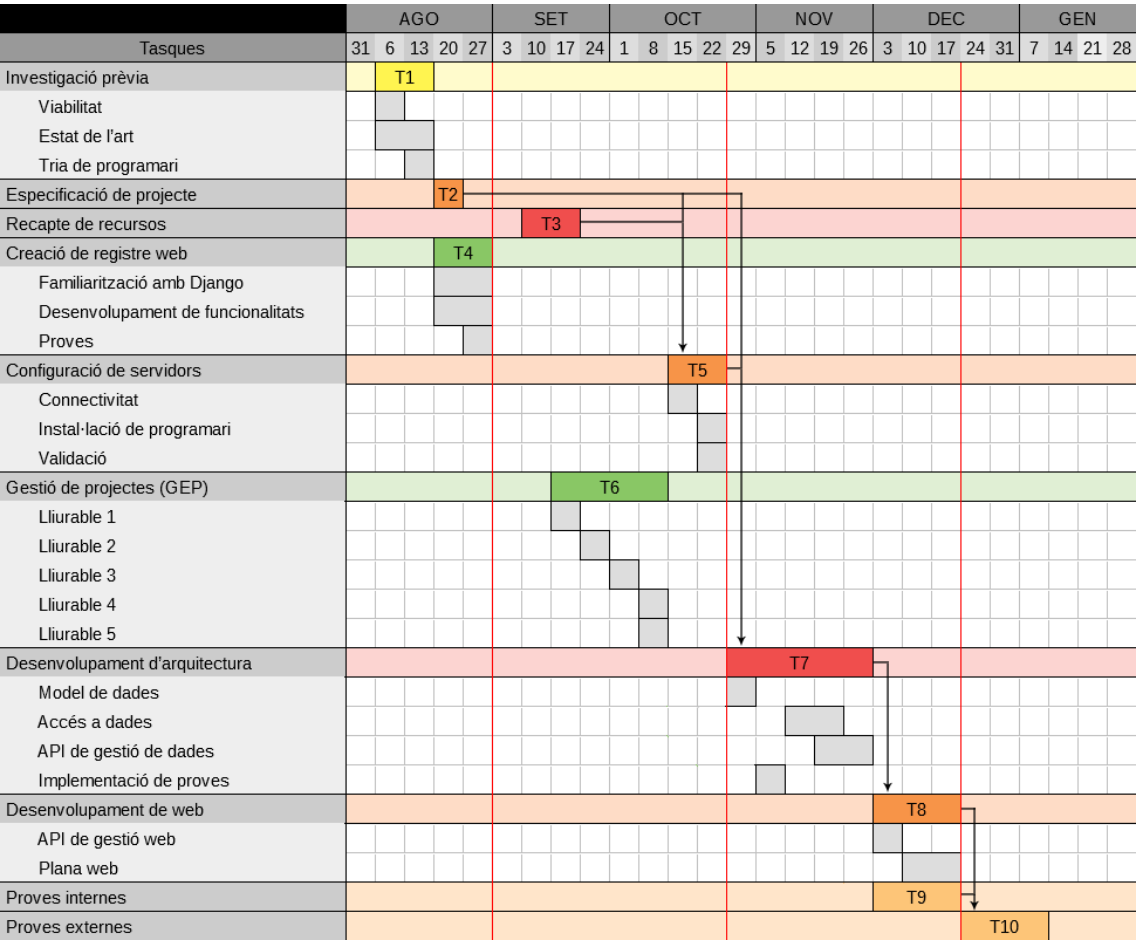


Figura 22: Esquema de Gantt representatiu de la planificació en fer la fita de seguiment i fita final.

B Diagrames de flux

Aquest annex desglossa en diagrames de flux les operacions que es realitzen per a cada un dels casos d'ús de l'aplicació web. Les relacions entre els diferents programaris usats es troba indicada mitjançant connexions direccionals, que es representen amb un traçat discontinu quan la comunicació comporta la transmissió de dades pròpies de l'aplicació.

B.1 Creació d'un torneig

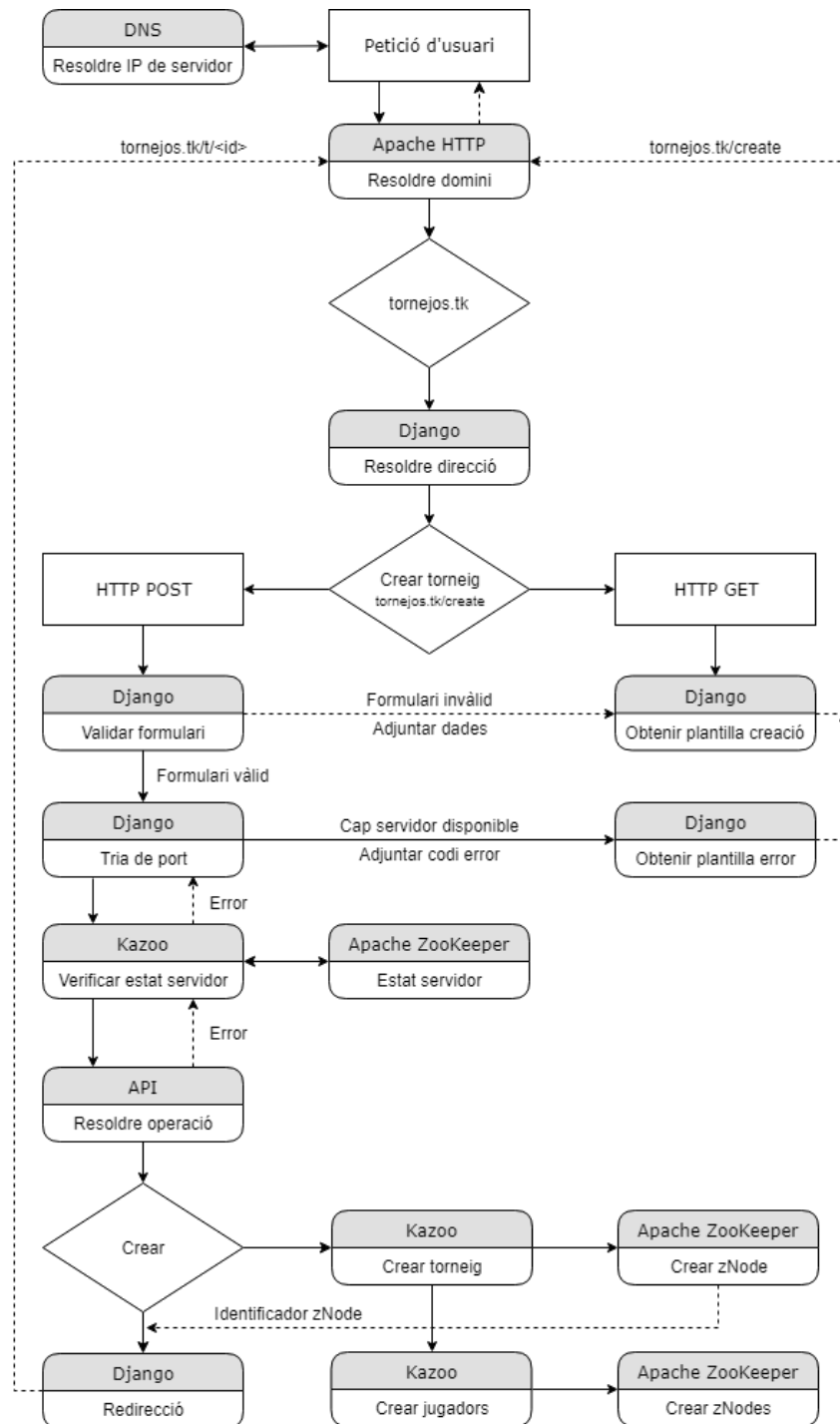


Figura 23: Diagrama de seqüència per a la creació d'un torneig.

B.2 Llistat de tornejos creats

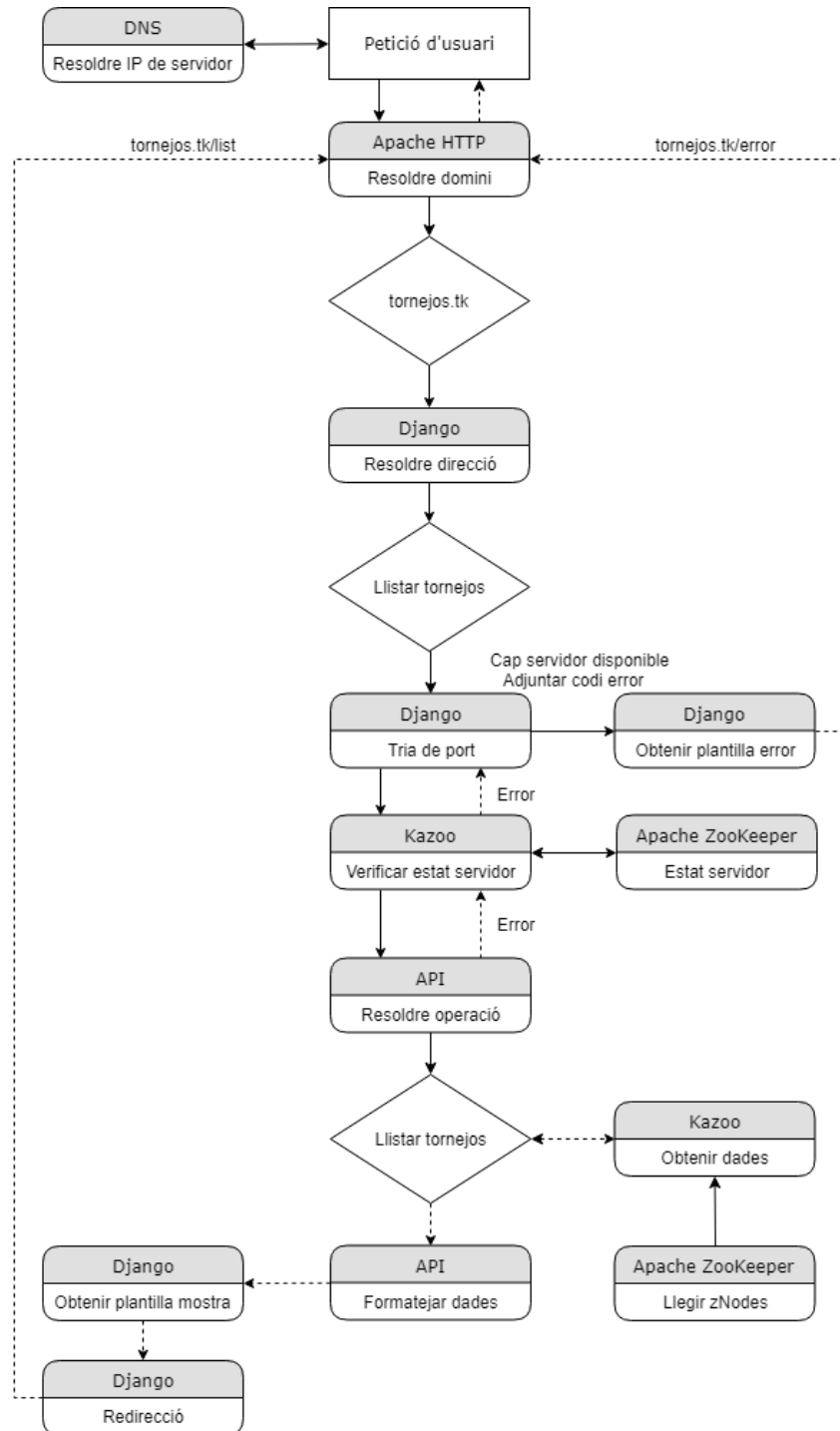


Figura 24: Diagrama de seqüència per al llistat de tornejos creats.

B.3 Consulta d'un torneig

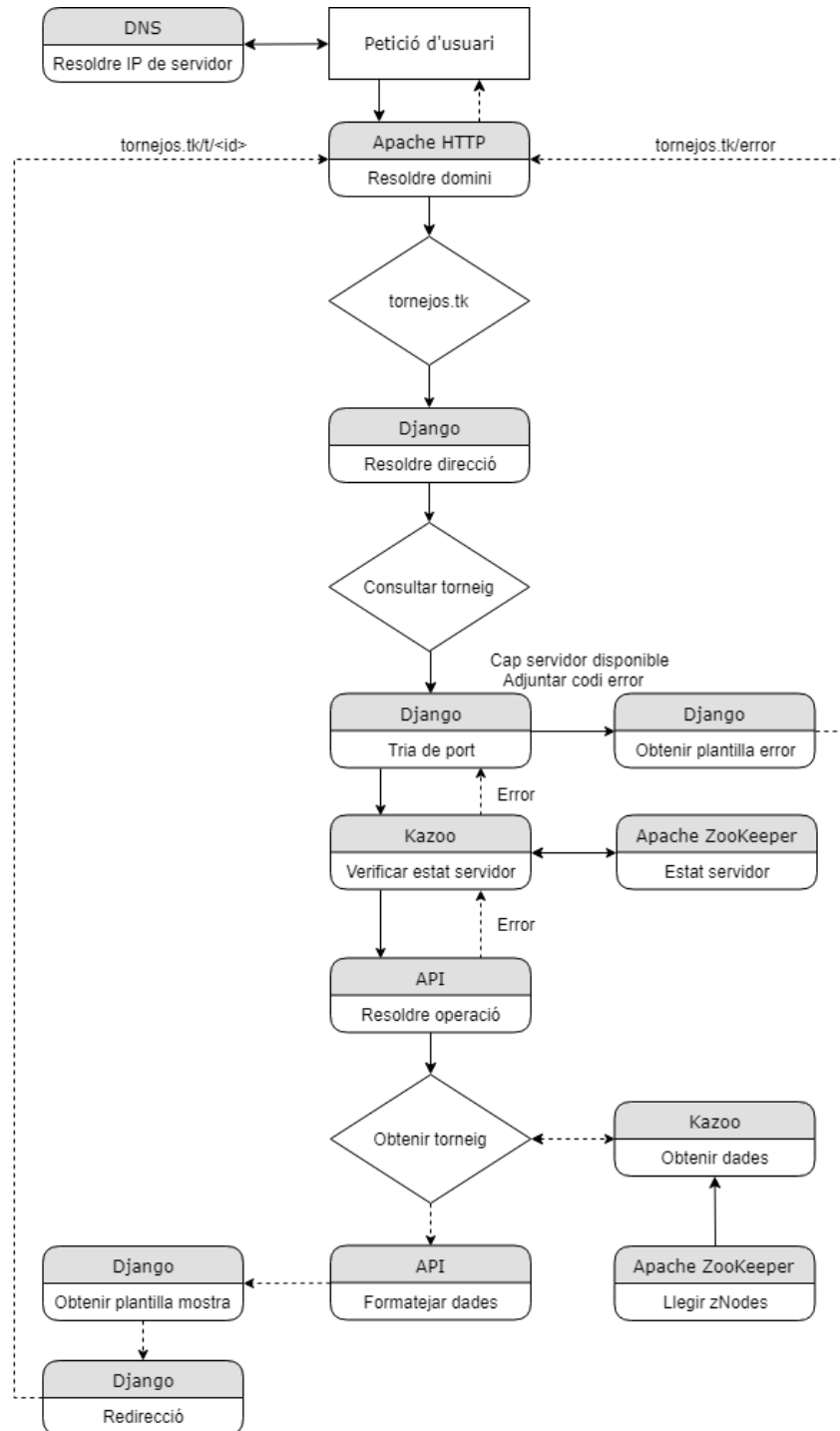


Figura 25: Diagrama de seqüència per a la consulta de la classificació en un torneig.

B.4 Actualització o eliminació de dades

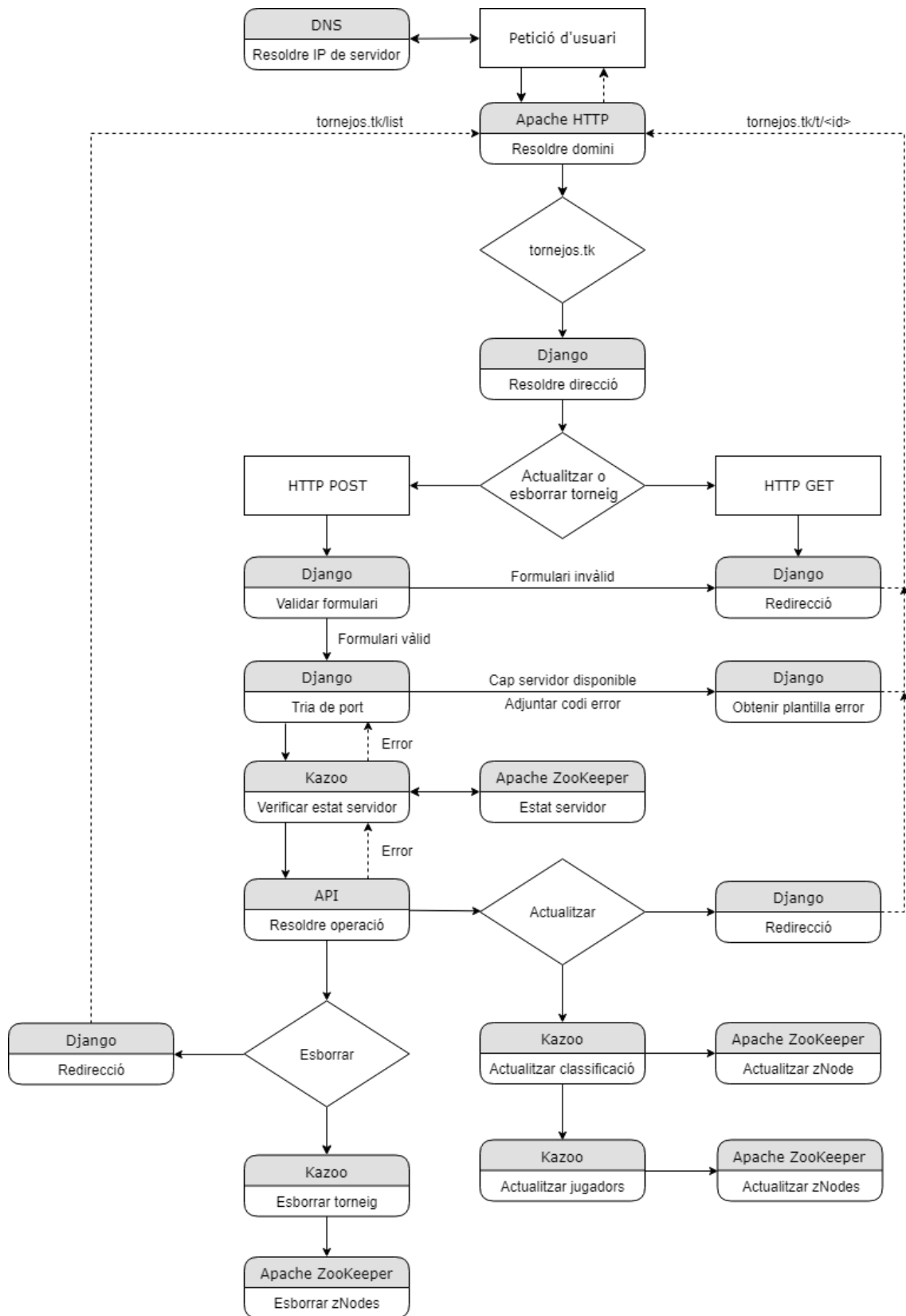


Figura 26: Diagrama de seqüència per a la modificació o eliminació d'un torneig.

C Codi

C.1 Instal·lació d'Apache ZooKeeper

Aquest *script* escrit en llenguatge Bash permet instal·lar Apache ZooKeeper a qualsevol servidor Unix, resolent i instal·lant les dependències necessàries i configurant les variables d'entorn.

Com es pot veure a l'inici del script (Línies 5 a 8), cal definir certes variables de forma manual per a configurar correctament l'eixam de servidors. Les variables indicades tenen com a finalitat identificar les direccions IP de tots els servidors de l'eixam, i l'identificador del servidor des del qual s'està usant el script. Si es volgués configurar l'eixam per a acceptar més servidors, caldria afegir les variables adients a continuació de la línia 8, i repetir les línies 38 a 40 amb les noves variables i incrementant l'identificador dels servidors.

Per tal d'iniciar el script es pot usar un intèrpret de Bash; un cop finalitzada la instal·lació apareixerà un directori anomenat *zookeeper* i un altre script anomenat *launch-zookeeper.sh*. El directori conté tota la configuració i els registres d'activitat del servidor, i el script té com a finalitat poder iniciar ZooKeeper al servidor local. Per a iniciar l'eixam correctament cal usar el nou codi a tots els servidors de l'eixam. Aquesta operació es pot automatitzar afegint l'entrada corresponent a un sistema de tasques com pot ser *cron* als servidors Unix.

```
1  #!/bin/bash
2  #####
3  # Variable definition #
4  #####
5  ip1="192.168.1.14"
6  ip2="192.168.1.15"
7  ip3="192.168.1.16"
8  myid=1
9
10 #Download and unzip zookeeper
11 echo "
12 -----
13   Downloading Apache Zookeeper from repository
14 -----"
15 #   Download
16 curl http://ftp.cixug.es/apache/zookeeper/stable/zookeeper-3.4.12.
17   tar.gz -o zookeeper-3.4.12.tar.gz
18 #   Unzip
19 echo "-----
20   Unzipping contents - Be patient
21 -----"
22 tar -zxvf zookeeper-3.4.12.tar.gz
23 #   Rename
24 mv zookeeper-3.4.12 zookeeper
25 #   Delete zip
26 rm zookeeper-3.4.12.tar.gz
27 echo "Unzip OK!"
28
29 #Set up config file
30 echo "-----
31   Setting up zookeeper config
32 -----"
33 echo "tickTime=2000
34   initLimit=10
35   syncLimit=5
36   dataDir=$PWD/zookeeper/data
37   clientPort=2181
38   server.1=$ip1:2888:3888
```

```

39 server.2=$ip2:2888:3888
40 server.3=$ip3:2888:3888
41 " > zookeeper/conf/zoo.cfg
42 #Create data dir
43 mkdir zookeeper/data
44 #Set up myid
45 echo "$myid" > zookeeper/data/myid
46
47 echo "Config OK"
48
49 #Ensure we have a JDK installed, or install version JDK v8
50 echo "-----"
51 Setting up java config
52 -----"
53 javac -version || ( yes | sudo apt-get update && yes | sudo apt-get
    install openjdk-8-jdk )
54
55 #Find jdk installation folder, usually /usr/local/jvm/<jdk>
56 javaout=$(which javac)
57 javahome=$(sudo readlink -f $javaout | sed -s 's/\/bin\/javac//')
58
59 #Ensure no errors during installation or exit
60 if [ -z $javahome ] ; then
61     echo "Unable to find the JDK file in the system. Aborting!"
62     exit -1
63 fi
64 echo "Java JDK found in $javahome"
65 echo "Config OK"
66
67 #Add JAVA_HOME var to bashrc
68 echo "-----"
69 Setting JAVA_HOME in .bashrc
70 -----"
71 echo "
72 JAVA_HOME=\$JAVA_HOME:$javahome
73 export JAVA_HOME" >> /home/$USER/.bashrc
74 #Refresh source
75 source /home/$USER/.bashrc
76 echo "Java home:
77 $JAVA_HOME"
78
79 echo "cd zookeeper && java -cp zookeeper-3.4.12.jar:lib/slf4j-api
    -1.7.25.jar:lib/slf4j-log4j12-1.7.25.jar:lib/log4j-1.2.17.jar:
    lib:conf org.apache.zookeeper.server.quorum.QuorumPeerMain $PWD/
    zookeeper/conf/zoo.cfg" > $PWD/launch-zookeeper.sh
80
81 echo "
82 #####
83 # INSTALLATION FINISHED #
84 #####
85 Launch Apache Zookeeper using the following line:
86
87 bash launch-zookeeper.sh"

```

C.2 Inspecció de dades

Aquest *script* en llenguatge Python permet inspeccionar les dades emmagatzemades a un servidor amb Apache ZooKeeper. És d'especial utilitat per a verificar el comportament del sistema i observar quin tipus de dades s'emmagatzemen a l'eixam. Una característica que s'ha afegit al codi és, mitjançant el pas d'una IP com a paràmetre, la possibilitat de redirigir les peticions cap a un altre servidor que tingui el programari d'Apache Zookeeper instal·lat i iniciat. L'ús d'aquest script és completament segur, ja que les connexions als servidors només es poden realitzar des de la xarxa privada a la qual pertanyen. Per tant, encara que la sortida del script mostri els mots de pas de cada torneig, no es comprometen les dades dels usuaris.

```
1 import sys
2 import logging
3
4 from kazoo.client import KazooClient
5
6 def print_node(client, nodename, depth):
7     """Print the information stored in a node."""
8     #Add spacing to match depth
9     print('\t'*depth, end='')
10    #Retrieve data
11    data, stats = client.get(nodename)
12    print("{0}".format(nodename), end='')
13    print("\t>>>\t\"{0}\"".format(data)) if data else print()
14    #Check for subnodes
15    if stats.children_count:
16        childrenNames = client.get_children(nodename)
17        for cname in childrenNames:
18            print_node(client, nodename + "/" + cname,
19                        depth+1)
20
21 def print_tree(client, host):
22     """Print the data tree."""
23     print("Displaying stored zNodes in tree structure ({})".
24           format(host))
25     print_node(client, '', 0)
26
27 if __name__ == '__main__':
28     #Default to localhost
29     host = '127.0.0.1'
30     port = '2181'
31     if len(sys.argv) > 1:
32         #Query specified host
33         host = sys.argv[1]
34     #Launch client
35     zk = KazooClient(hosts=':'.join([host, port]))
36     zk.start()
37     #Print tree data
38     print_tree(zk, host)
39     zk.stop()
40     zk.close()
```


C.3 API de comunicació

Client

La part de client de l'API de comunicació és l'encarregada d'enviar les peticions als diferents servidors de l'API; en el cas del projecte implementat les funcions del client es criden des de l'entorn de Django. L'extracte de codi llistat a continuació és l'encarregat de la tria dels ports i verificació de disponibilitat dels diferents servidors. En cas que no hi hagi cap servidor disponible es retorna un codi d'error (-1) amb un missatge indicatiu.

```
1  (...)
2  def __get_listed_ports():
3      listed_ports = cache.get('ports')
4
5      if not listed_ports:
6          # Ports not in cache. Read from file
7          client_ports = ''
8          with open('/home/pi/tfg/scripts/listeners.cfg', 'r'
9              ) as f:
10              lines = f.readlines()
11
12              start_port = int(lines[0].rstrip("\n"))
13
14              zookeeper_clients = lines[1].rstrip('\n')
15              cache.set('zoo_clients', zookeeper_clients,
16                  None)
17              client_count = len(zookeeper_clients.split(
18                  ","))
19              client_ports = range(start_port, start_port+
20                  client_count)
21
22              listed_ports = ",".join([str(n) for n in list(
23                  client_ports)])
24              cache.set('ports', listed_ports, None)
25      return listed_ports
26
27 def __get_port(tried_ports):
28     """Get a listening port from the available list. """
29     listed_ports = __get_listed_ports()
30     # Filter tried ports
31     available_ports = []
32     for port in listed_ports.split(","):
33         if port not in tried_ports:
34             available_ports.append(port)
35
36     # Select a random port
37     if available_ports:
38         return random.choice(available_ports)
39     else:
40         return None
41
42 def __send_req_to_api(data):
43     tried_ports = []
44     port = __get_port(tried_ports)
45
46     while port is not None:
47         address = ("localhost", int(port))
48         result_data = ''
```

```

45         try:
46             client = Client(address)
47             client.send(data)
48             # Wait 0.75s for a response, or change
               server
49             if client.poll(0.75):
50                 result_data = client.recv()
51                 client.close()
52         except ConnectionRefusedError:
53             # Client not available, try next
54             pass
55         else:
56             if result_data:
57                 code = result_data['code']
58                 if code is not None and code != -1:
59                     # Connection to zookeeper
                       OK
60                     return result_data
61
62             # Connection failed, get new port
63             tried_ports.append(port)
64             port = __get_port(tried_ports)
65
66     if port is None:
67         return {'code': -1, 'data': "No nodes available"}
68
69     (...)

```

Servidor

La part de servidor de l'API de comunicacions és l'encarregada de rebre les peticions que realitzen els diferents clients. Quan es rep una petició, s'analitza quina operació es vol realitzar i es comprova que tingui els paràmetres adients. En cas que faltin paràmetres, l'operació sigui invàlida o hi hagi un error durant l'execució de la operació sol·licitada, el servidor indica als clients que no es pot completar l'operació sol·licitada amb un codi d'error i adjunta un missatge indicatiu.

```

1  import sys
2  import re
3
4  from multiprocessing.connection import Listener
5  from kazoo.client import KazooClient
6  from kazoo.client import KazooState
7
8  from data_api import DataAPI
9
10 class DataApiListener():
11
12     def __init__(self, pipe_address, zoo_address):
13         # Requests pipe
14         self.listener = Listener(pipe_address)
15         # Kazoo client
16         self.kz_client = KazooClient(hosts=zoo_address)
17         self.kz_client.start()
18         # Requesting API
19         self.__api = DataAPI(self.kz_client)
20         # Flag to disconnect
21         self.__keep_running = True

```

```

22
23
24     def run(self):
25         """Run a daemon, accepting requests for the API.
26
27         This daemon will run until stop() is called on the
28         ApiListener instance.
29         """
30         while self.__keep_running:
31             new_c = self.listener.accept()
32             # Check there is data to read
33             if not new_c.poll(0.2):
34                 # Close after 0.2 seconds idle
35                 new_c.close()
36                 continue
37             # Accept new connections
38             c_data = new_c.recv()
39             # Request api
40             result = self.__parse_request(c_data)
41             # Return result of operation
42             new_c.send(result)
43             # Close connection
44             new_c.close()
45
46             # Close socket
47             self.listener.close()
48             # End Kazoo connection
49             self.kz_client.stop()
50             self.kz_client.close()
51
52
53     def stop(self):
54         """Stop further connections to this client."""
55         self.__keep_running = False
56
57
58     def __parse_request(self, req):
59         """Parse an incoming request.
60
61         Returns a dictionary containing the following items
62         res = {
63             code : <integer>,
64             data : <string, dict>
65         }
66         """
67         result = {'code' : 0, 'data' : ''}
68
69         if self.kz_client.state is not KazooState.CONNECTED
70             :
71             result['code'] = -2
72             result['data'] = "Server unavailable"
73             return result
74
75         if type(req) is not dict:
76             result['code'] = -1
77             result['data'] = "Invalid data format"
78             return result

```

```

79         try:
80             result['data'] = self.__run_request(req)
81         except Exception as e:
82             result['code'] = -3
83             result['data'] = str(e)
84
85         return result
86
87
88     def __run_request(self, req):
89         result = ''
90         op = req.get('operation', None)
91         data = req.get('data', None)
92         if op == "create":
93             result = self.__api.create_tournament(
94                 name = data['name'],
95                 modality = data['modality'],
96                 password = data['password'],
97                 players = data['players']
98             )
99         elif op == "update":
100             result = self.__api.update_tournament(
101                 tournament_id = data['id'],
102                 version = data['version'],
103                 classification = data['
104                     classification'],
105                 password = data['password']
106             )
107         elif op == "delete":
108             result = self.__api.delete_tournament(
109                 tournament_id = data['id'],
110                 password = data['password']
111             )
112         elif op == "get":
113             result = self.__api.get_tournament(
114                 tournament_id = data['id']
115             )
116         elif op == "get_list":
117             result = self.__api.get_tournament_list()
118         elif op == "status":
119             result = {
120                 'status' : self.kz_client.state,
121                 'address' : self.kz_client.hosts
122             }
123         elif op == "setpath":
124             new_path = data['path']
125             # Match filesystem format "/path/to/file"
126             if re.compile("(/[\\d\\w_]+/?)*").match(
127                 new_path):
128                 self.__api.set_data_path(data['path
129                     '])
130                 result = data['path']
131             else:
132                 raise Exception("Malformed path")
133         elif op == "dummy":
134             result = 'OK'
135         else:
136             raise Exception("Operation " + op + " is

```

```
134             invalid.")
135         return result
136
137 if __name__ == '__main__':
138     if len(sys.argv) != 3:
139         print("Wrong argument count!")
140         print("python3 {0} <socket_port> <zookeeper_ip:port>".format(
141             sys.argv[0]))
142     else:
143         pipe_address = ("localhost", int(sys.argv[1]))
144         zoo_address = sys.argv[2]
145         listener = DataApiListener(pipe_address,
146             zoo_address)
146         listener.run()
```